

תוכן מסמך זה הוא דוגמא
טובה לספר פרויקט, תמיד
יש מה לתקן.

יש להפעיל שיקול דעת
ולתאים ספציפית לפרויקט
בהתאם להנחיות הניתנות
בנוהל הפרויקטים העדכני
לאותה שנה ובהתאם
להנחיות המנחה של
הפרויקט.



אפקה המכללה האקדמית להנדסה בתל-אביב
AFEKA TEL-AVIV ACADEMIC COLLEGE OF ENGINEERING

מחלקת הנדסת תוכנה

שם הפרויקט: פורטל אומנות דיגיטלית

Project Name: Digital Art Portal

ספר פרוייקט

שם הסטודנט:

שלום יוני

מספר תעודת זהות:

0-3909419-8

שם המנחה:

גדעון קור

תאריך ההגשה:

15 / 05 / 2008

חתימת המנחה:

1. פתיחה

1.1. תקציר מנהלים

המונח "אומנות דיגיטלית" מתייחס ליצירות שעובדו או נוצרו על ידי מחשב, באופן דיגיטלי. מקור היצירה יכול להיות בתוכנת מחשב בלבד, כמו פרקטלים, אומנות אלגוריתמית, או לקוחה ממקור אחר- כגון תמונה סרוקה, או ציור שנוצר באמצעות גרפיקה מבוססת וקטורים (שילוב בין יצירה ידנית לתוכנת מחשב).

למרות שלעיתים המונח נמצא בשימוש במושג הרחב שלו – כלומר – מיוחס גם לתמונות שפשוט נסרקו למחשב, בדרך כלל ההגדרה שמורה לאומנות אשר שונתה באופן מהותי או נוצרה בלעדית על ידי תהליך מחשבי כלשהו – כגון תוכנת מחשב, מיקרו-מעבדים או מערכת אלטרונית אחרת שמפענחת את הקלט הדיגיטלי ומעבדת אותו באופן כלשהו ליצירת פלט חדש.

כיום, הרוב המוחלט של מגמות האומנות באקדמיה, מכילות או מובילות לעסיקה בתחום זה, בניגוד לתחום האומנות ה"קלאסי"- שכן הדרישה מצד התעשייה מתמקדת בעיקר ביכולות גבוהות בעיבוד ויצירה קריאטיבית של תוצרים ויזואליים למטרות שיווק, דפוס או בידור.

החוג האקדמי הכולל בין כתליו התמחויות בעיצוב גרפי ומגמות חופפות, כולל כיום כ-16 מרכזים אקדמיים ומכללות, הכוללות את המכון הטכנולוגי חולון, בצלאל, האקדמיה לעיצוב ויצו חיפה, ועוד.

יחד עם זאת, ישנו פער בין הארץ לעולם בכמות, איכות וגודל הקהילות המקוונות הקשורות לתחום. לעומת אתרי-ענק כוללניים לתמונות כגון Flickr, אתרי אומנות, באופן טבעי, יותר ממוקדים הן בתוכן והן בפונקציונליות לקהל היעד, ומהווים במת ביטוי והחלפת רעיונות מקצועית נוספת, במקביל לאקדמיה ולתעשייה.

פרוייקט הגמר הינו מערכת תוכנה, אשר מהווה בסיס לקהילה מקוונת בתחום האומנות הדיגיטלית. המטרה העיקרית של המערכת הינה לאפשר את קידום האישי של אומנים צעירים בתחום הגרפיקה הממוחשבת, בין אם לצורך שיפור תיק העבודות לקראת קבלה למוסד לימודים, או לצורך התמודדות על משרה.

קהל היעד העיקרי של המערכת הוא סטודנטים לעיצוב חזותי, "ביצועיסטים" בתחום, חובבי צילום ומתעניינים.

המערכת משלימה חוסר אמיתי וקיים בארץ ובעולם, במתן אפשרות לאומנים ועוסקים בתחום לשתף פעולה ולמטב יכולותיהם אחד של השני, יחד עם הגברת החשיפה של התחום והחדרתו למודעות.

הצרכים העיקריים עליהם באה לענות המערכת הינם :

- מתן דגש על פיתוח אישי של היוצר
- שיפור ממשק המשתמש ביחס למערכות אחרות המוצעות כיום
- הצעת יכולות חדשות שלא קיימות באתרים הנוכחיים, והכנסת אלמנטים "תחרותיים"

המערכת הינה אתר אינטרנט המקושר לתחום הפנאי והבידור, וככזה אינו בא לענות לצורך ארגוני או ניהולי כלשהו. המדד הפרקטי להצלחת המערכת (במסגרת פרוייקט הגמר) הינו עמידה בדרישות שהוגדרו. המדדים התיאורטיים להצלחת המערכת הינם מדדים של רישום משתמשים ביחס לחשיפה, ועל פי כמות הפעילות השוטפת באתר.

הרכיבים העיקריים בעלי עניין בעבודה הם שכבת ממשק המשתמש, ורכיב ה-Media Server אשר מהווה שרת אחסון, רכיב עיבוד תמונה וניהול קבצי מדיה. זהו רכיב נפרד בעל ממשק אפליקטיבי אשר מסוגל לשרת מספר אפליקציות, אשר פרוסות על גבי מספר שרתים.

השיטות בהן פותחה המערכת משלבות טכנולוגיות חדשניות בתחום תשאול המידע וגישה למידע, וטכנולוגיות מבוססות ומוכחות בעולם מערכות ה-WEB, כאשר דגש וחשיבה הושמו בממשק המשתמש, מתן בסיס ארכיטקטוני טוב להרחבה ותמיכה במספר משתמשים מרובה, וקלות תחזוקה.

1.1. Executive Summary

The term "digital art" is associated with artwork which has been created or processed via a computer system, in a digital manner. The source of the artwork can be a computer software system alone, like fractals, algorithmic art, etc. – or other sources, such as a computer-scanned image, or a vector based drawing created with graphics software and a drawing tablet (combination between manual, or – "classical" art, and computer software and hardware systems).

Although sometimes the term "digital art" is used in a wider context, meaning – is also attributed to scanned images or drawings, it is usually reserved to artwork which has been significantly altered or solely manufactured using some kind of computer-assisted process or computer-based component, such as computer software, micro-processors or other electronic system which is capable of deciphering input and processing it in order to create new output.

Nowadays, the vast majority of academic art and visual design programs employ techniques used in, or lead to vocation in this field, since the industry dictates an ever growing demand for highly skilled individuals with distinctive capabilities in performing high quality image manipulation and other creative work in order to produce visual products for marketing, print or entertainment purposes.

Still, there is a gap in Israel and the rest of the world in the quality, availability and size of online communities focused on this area of industrial expertise. Unlike giant image-storing websites such as Flickr.com, art websites are naturally inclined to place much more focus on both content and functionality which is relevant to the art target audience, and are a significant alternative stage of self expression and exchange of ideas, in parallel to the academy and the industry.

This final project is a software system, which is meant to provide a basis for such an online community. The main goal of this system is to enable the personal growth and advance of young and aspiring artists in the fields of visual art and design, whether it's in order to improve their personal portfolio, and with it – their chances of acceptance to various academic institutions, applying for a job, on top of being an impressive online gallery of artwork, which brings the fun element into play.

The target audience for this system is visual design students, amateur performers, art fans and actually – anyone interested in the field.

The system aims to fill a true gap in Israel and the world, by allowing artists and other professionals to collaborate and share opinions and ideas better than before, and to intrinsically increase awareness of this field.

The main goals for this system are:

- Emphasizing personal growth and evolution of the artist.
- Improving the user interface in comparison to current online systems.
- Offering new and unique capabilities and features, and introducing competitive elements into play.

The system is an online web site which is related to the fields of leisure and entertainment, and as such – does not come to answer any organizational or business needs. The practical measure for success for the system, within the limitations of the final project's resources and time limits, is the accomplishment of the requirements as depicted in system requirements documentation.

The theoretical measures for success include statistical inputs, mainly marketing related, which would primarily account for user interaction facts such as registration-to-exposure rate, and other activities being performed on the website.

The main components of interest in this system are the UI layer, and the Media Server component which functions as a storage server, image processing component and media file management systems. This sub system is separate from the main application process, which is capable of, and designed to, support for multiple applications, deployed on multiple server instances.

The methodologies used while producing the system include innovative technologies in the field of data querying and retrieval, and classical time-tested technologies in the world of web systems, whilst putting emphasis and much thought to the user interface, and providing with a good foundation for further scalability and multi-user support.

1.2. תוכן עניינים

1.	פתיחה	3
1.1.	תקציר מנהלים	3
1.2.	Executive Summary	4
1.3.	תוכן עניינים	6
1.4.	רשימות	8
1.5.	מילון מונחים	8
1.6.	מבוא	9
1.7.	מטרות ויעדי הפרוייקט	9
1.8.	סקירה ספרותית	11
1.9.	תאור מצב קיים כולל ניתוח חלופות מערכתי	11
2.	דרישות המערכת (Software Requirements)	19
3.	אפיון המערכת (Software Specifications)	20
1.1.	מודל המערכת	20
1.2.	אפיון פונקציונלי	22
1.3.	ביצועים עיקריים	23
1.4.	אילוצי סביבה	23
1.5.	ניתוח חלופות טכנולוגיות	23
4.	תיכון המערכת	27
1.1.	ארכיטקטורת המערכת	27
1.2.	תיכון מפורט	28
1.3.	אלטרנטיבות לתיכון	29
5.	תכנון הפרוייקט	31
1.1.	ניהול סיכונים	31
1.2.	תוכנית עבודה	31
6.	בדיקות והערכה (Software Testing and Evaluation)	31
1.3.	תוכנית בדיקות תוכנה	31
1.4.	דווח בדיקות תוכנה	31
1.5.	דוגמאות הפעלה מפורטות מקצה לקצה	32
1.6.	ניתוח יעילות	32
1.7.	ביצועי מערכת	38
1.1.1.	עלות מערכת	38
1.1.2.	אמינות (איכות) המערכת	38
1.1.3.	שלמות המערכת	39

39	1.1.4. ייחודיות ומקוריות
39	1.8. אבטחת מידע
39	7. התוצר
43	8. סיום
43	1.9. סיכום ומסקנות
43	1.10. פיתוחים עתידיים והמשך עבודה:
43	1.11. רשימת מקורות.

1.3. רשימות

12.....	דוגמה 1
12.....	דוגמה 2
13.....	דוגמה 3
18.....	טבלה 1
20.....	איור 1 – המערכת במבט על
21.....	איור 2 - ERD
22.....	איור 3 - DFD
27.....	איור 4 - ארכיטקטורה

1.4. מילון מונחים

CLR – Common Language Runtime
BCL – Base Class Library
DLinQ - Language Integrated Query for Data
Linq – Language Integrated Query
ORM – Object Relational Mapping
SOA – Service Oriented Architecture
WCF – Windows Communication Foundation

1.5. מבוא

הפרויקט הינו חלק ממסלול הלימודים לתואר ראשון בהנדסת תוכנה B.Sc. במכללה האקדמית להנדסה בתל אביב. הפרויקט הינו מערכת WEB אשר מיועדת לקהל הרחב. משתמשי המערכת הם סטודנטים לעיצוב חזותי, "ביצועיסטים" בתחום, חובבי צילום ומתעניינים.

כותב הפרוייקט הוא יוני שלום, סטודנט להנדסת תוכנה ומתכנת בחברת MyThings.com, נכון ליום כתיבת מסמך זה.
מנחה הפרוייקט הוא מר גדעון קוך, יזם ומרצה.

ברצוני להודות לגברת יונית שוורץ ומר גדעון קוך אשר שימשו כמנחי פרוייקט זה בשלביו השונים, ולחברת MyThings.com אשר תספק את שירותי האחסון למערכת בשלביה הראשונים.

1.6. מטרת ויעדי הפרוייקט

קהל היעד העיקרי של הפורטל הוא סטודנטים לעיצוב חזותי, "ביצועיסטים" בתחום, חובבי צילום ומתעניינים. ישנם כיום מספר הולך וגדל של מסלולי תואר ראשון ותעודה בתחום התקשורת החזותית, במוסדות כגון האוניברסיטה הפתוחה ("חשיפה"), מכללת תילתן, מכללת ויצ"ו חיפה, המכללה האקדמית ספיר, המרכז האקדמי בצלאל, והמכון הטכנולוגי בחולון. המערכת המוצעת משלימה חוסר אמיתי וקיים בארץ ובעולם, במתן אפשרות לאומנים ועוסקים בתחום לשתף פעולה ולמטב יכולותיהם אחד של השני, יחד עם הגברת החשיפה של התחום והחדרתו למודעות.

מעבר לכך, חדירה מוצלחת של מערכת כזו מביאה לאלמנטים תחרותיים אשר יסייעו להעלות את הרמה המקצועית האישית של רבים.

המטרה העיקרית של המערכת המוצעת הינה להביא לתחום יכולות ופונקציונליות נוספת אשר תאפשר למנף את יכולות המערכת לטובת קידום האיש של המשתמשים בתחום, בין אם לצורך שיפור תיק העבודות לקראת קבלה למוסד לימודים, או לצורך התמודדות על משרה.

המטרות העיקריות של המערכת הן :

- מתן במה לביטוי אישי של יוצרים בתחום האומנות הדיגיטלית
- שיפור ממשק המשתמש
- הצעת יכולות חדשות שלא קיימות באתרים הנוכחיים, והכנסת אלמנטים "תחרותיים"

המערכת הינה אתר אינטרנט המקושר לתחום הפנאי והבידור, וככזה אינו בא לענות לצורך ארגוני או ניהולי כלשהו. המערכת תוערך על הצלחתה על פי מדדים של רישום משתמשים ביחס לחשיפה, ועל פי כמות הפעילות השוטפת באתר.

כיום קיימות מספר מערכות דומות, כגון DeviantArt.com העולמית, ו-stage.co.il או cgart.co.il הישראליות. המטרה העיקרית של המערכת הינה לשפר את ממשק וחווית המשתמש, ולהביא לתחום יכולות נוספות אשר תאפשר למנף את יכולות המערכת לטובת קידום האיש בתחום, בין אם לצורך שיפור תיק העבודות לקראת קבלה למוסד לימודים, או לצורך התמודדות על משרה.

היעד הפרקטי העיקרי של הפרוייקט הוא הקמת מערכת פעילה אשר תהווה בסיס לקהילה מקוונת בתחום האומנות הדיגיטלית, הכוללת את הפונקציונליות המתוארת בדרישות.

מדדי הצלחה תיאורטיים :

- אחוז הנרשמים מתוך המשתמשים שנחשפו לאתר. (מעל 20%)
- אחוז המשתמשים החוזרים פעם שניה, מתוך אלו שנרשמו. (מעל 40%)
- אחוז המשתמשים הפעילים באתר (2 כניסות יחודיות) מתוך אלו שנרשמו. (מעל 1000 כניסות יחודיות ביום)
- כמות הכניסות היומית לאתר ביחס לגודל קהל היעד המוערך (מעל 40% מקהל היעד)

(הערה : עקב העובדה שהאתר לא יהיה חשוף לאינטרנט במסגרת הקמת הפרוייקט ולא יבוצעו פעילויות שיווק, מדדי הצלחה אלו הינם תיאורטיים בלבד).

ניתן לבחון מטרות ויעדים סופיים הנ"ל אל מול אלו אשר צוינו בהצעת הפרוייקט המורחבת, נספח שישי, סעיף 2.

1.7. סקירה ספרותית

1.7.1. NET and the CLR

המושג .Net. הינו מותג מסחרי אשר הוטבע בשנת 2002 על ידי חברת מיקרוסופט, עם השקת גירסה 1.0 של ה- .Net Framework. - תשתית פיתוח וריצה לתוכנה, אשר הינה אוסף של יחידות תוכנה ותשתית. אלו מאפשרות לשלב יכולות ייחודיות, הן לתהליך הפיתוח והן למוצר התוכנה עצמו.

על מנת לפתח מוצר תוכנה איכותי על גבי תשתית זו, ישנו צורך להכיר את השירותים אותם היא מציעה, העקרונות על פיהן היא פועלת, ופרדיגמות הפיתוח אותן היא ממשת. דוגמאות ממשיות לצורך זה באות לידי ביטוי בעיקר בשיקולי ביצועים ובעלות זמן הפיתוח.

ה (CLR (Common Language Runtime) הינו אחד מאבני הבסיס של תשתית זו. רכיב זה הוא למעשה סביבת הריצה – הוא מנהל את ריצת התוכנה, מבצע פעולות תשתית חיוניות כגון ניהול זיכרון, חלוקת הקוד לקטעי ביצוע וכדומה. שפות הכתובות תחת תשתית .Net אינן מומרות ישירות לשפת מכונה בעת הקומפילציה – אלא לשפה הנקראת IL (Intermediate language), אשר מומרת לשפת מכונה בזמן הריצה על ידי ה-CLR.

במהלך פרוייקט זה אני מביא לידי שימוש מספר יכולות מפתח של תשתית זו, והעיקריות שבהן:

Generics

יכולת זו הוצגה ב- .Net Framework. החל מגירסה 2.0, ומאפשרת ליצור טיפוסים נתונים, פונקציות וממשקים גנריים, אשר מממשים התנהגות דומה אך פועלים על אובייקטים מסוגים שונים (בדומה ל- Templates ב- C++)

מפתחים אשר אימצו את השימוש ב- Object Oriented Programming -מודעים ליתרונות ששיטה זו מציעה. אחד היתרונות העיקריים הינו היכולת להשתמש באותו קטע קוד מספר פעמים, על מנת לפתור בעיות שונות. (לדוגמה – ירושה של פונקציונליות קיימת ממחלקה מסויימת, חוסכת את הצורך לכתוב את הקוד שוב).

מנגנון ה-Generics מאפשר לנו לקחת רעיון זה ולהרחיב אותו למבני נתונים ולאלגוריתמים באופן קל, נוח, ולהקטין את הסיכון והעלות בביצועים הכרוכים בהמרת טיפוסים. (Casting)

לדוגמה – אוסף של אובייקטים. באמצעות Generics, אנו יכולים להגדיר, $List<T>$, כאשר T הינו פרמטר אשר מאפשר לנו ליצור לציין איזה טיפוס נתונים מוכל באוסף (האם זה אוסף של מספרים, מחרוזות, או אובייקטים אחרים)

מחלקה זו מתוארת באופן חלקי בדוגמת הקוד הבאה -

דוגמה 1

```
public class IntList
{
    //...
    public void Add(int item) { }
    public int this[Int32 index] { get { } set { } }
    //...
}

//Usage -
IntList list = new IntList();
list.Add(32);

//Get the third element in the collection
int num = list[3];
//...
```

עד כאן המחלקה שלנו עובדת, אבל היא מספקת פונקציונליות רק עבור איברים מסוג `int`. מה נעשה כשנרצה לתמוך בכל סוג שהוא של אובייקט? אפשרות אחת מוצגת בקוד הבא -

דוגמה 2

```
public class ObjectList
{
    //...
    public void Add(object item) { }
    public object this[Int32 index] { get { } set { } }
    //...
}

//Usage -
ObjectList list = new ObjectList ();
list.Add(32);
list.Add("Text");

//Casting is bad for performance...
int num2 = (int)list[0];

//RUNTIME ERROR - Can't cast String to Int
int num2 = (int)list[1];
//...
```

אם נביט בדוגמה זו, ניתן לזהות שתי בעיות—

1. אוסף הנתונים שלנו אינו הומוגני (מכיל אובייקטים מסוגים שונים), עובדה שתקשה עלינו להשתמש במנגנונים כגון פולימורפיזם, הרצת אלגוריתמים כלשהם על האוסף (השוואה בין שני אובייקטים) וכדומה.

2. כדי להשתמש באובייקטים שבאוסף הנתון – עלינו לבצע המרה של טיפוס הנתונים. המרה זה מוסיפה תקורה לזמן ריצת התוכנית, וניתן לראות שהיא גם מסוכנת – אנו עלולים לטעות בקלות ולקבל שגיאת זמן ריצה.

באמצעות Generics, ניתן לפתור בקלות בעיות אלו – אנו יכולים להגדיר `GenericList<T>`, כאשר `T` הינו פרמטר אשר מאפשר לנו ליצור לציין איזה טיפוס נתונים מוכל באוסף (האם זה אוסף של מספרים, מחרוזות, או אובייקטים אחרים).

דוגמה 3

```
public class GenericList<T>
{
    //...
    public void Add(T item) { }
    public T this[Int32 index] { get { } set { } }
    //...
}

GenericList<int> intlist = new GenericList<int>();
intlist.Add(32);
//...
int num = intlist[1];
//...

GenericList<string> stringList = new GenericList<string>();
stringList.Add("Text");
//...
string str = stringList[3];
//...
```

דוגמה זה הינה דוגמה פשוטה מאוד לשימוש במנגנון ה Generics-שכן מנגנון זה מאפשר שימוש חכם ב Design Patterns -מורכבים וכו'. נושא זה מוצג לעומק בספר -

Jeffrey Richter (2006) – "CLR Via C#, Second Edition" (359)

Asynchronous Programming Model

באופן כללי, מודל הריצה של קוד כלשהו הוא סיקוונסיאלי, (Sequential) משמע – שורות קוד בתוכנית מבוצעות אחת אחרי השניה. לעיתים קרובות ניתן למצוא בהם פעולות מסויימות לוקחות זמן רב, ואם נמתין שהן יסיימו את ריצתן – נפגע בהיבט כלשהו של התוכנית .

לדוגמה - תוכנית פשוטה אשר, בין השאר, שולחת בקשות ברשת למחשב מרוחק וממתינה לתוצאת הבקשה. נניח שהקוד שנכתב שולח את הבקשה וממתין לאחר מכן, עד שהתשובה חוזרת .

התוצאה – עד שהתשובה תחזור (מה שיכול לקחת זמן מה, במיוחד אם הפעולה נכשלת) - ממשק המשתמש יקפא והמשתמש המבולבל לא יבין מה קורה .

באותה מידה אנו עשויים לפגוע בפונקציונליות קריטית אחרת של המערכת – שכן התוכנה כולה ממתינה ולא מתבצעת אף שורת קוד עד שהתשובה תחזור מבקשת הרשת.

לשם כך ישנם מנגנונים אסינכרוניים, ביניהם שימוש ב Multithreading, אשר מאפשרים לשאר המערכת להמשיך לתפקד בזמן שפונקציות "גזלניות" (בדרך כלל מבוססות I/O) מבוצעות או ממתינות לתשובה.

שימוש במנגנונים אסינכרוניים הינו אחד העקרונות העיקריים בעת בניית מערכות תוכנה הנדרשות לביצועים גבוהים ויכולת התרחבות (Scalability) מתוך הבנה זו, צוות ה CLR-במיקרוסופט החליטו למצות את הפוטנציאל הגלום בנושא ולעצב תבנית פתרון שתקל על המפתחים להשתמש ביכולות אלה. תבנית זו נקראת APM (Asynchronous Programming Model).

תבנית זו הינה הפשטה של רעיון הביצוע האסינכרוני, והוא בא לידי ביטוי במספר רב של מקומות בספריית המחלקות המובנית ב- .Net, כגון – כל מחלקה היורשת מ Stream - (FileStream, NetworkStream), מחלקות היורשות מ- WebRequest (FileWebRequest, HttpWebRequest) ועוד.

לב הרעיון בתבנית זו הינו השימוש בפונקציות BeginXxx() ו EndXxx()-כאשר Xxx מצוין את הפעולה שיש לבצע .

תבנית זו מציעה 3 דרכים שונות "לסיים" את הפעולה האסינכרונית –

1. Wait-Until-Done - קוראים ל () BeginXxx- אבל קוראים ל- EndXxx() לפני שהפעולה הסתיימה – במצב זה עד שהפעולה לא תסתיים – האפקט יהיה דומה לפעולה סינכרונית. במנגנון זה כדאי להשתמש אם אנו יודעים בוודאות גבוהה שקיים זמן עיבוד רב בין הקריאה לתחילת הפעולה ועד לקריאה לסיומה.
 2. Polling - מבצעים בדיקה באינטרוול כלשהו, לבדיקה האם הפעולה הסתיימה. שיטה זו היא די בזבזנית שכן אנו מנצלים זמן ביצוע של Thread מסויים על הבדיקה.
 3. Callback-Method - בשיטה זה אנו מספקים למנגנון מצביע לפונקציה שלנו (Delegate), אשר ה- APM יקרא לו בעת סיום הפעולה האסינכרונית. זוהי השיטה היעילה והמומלצת ביותר לביצוע פעולות אסינכרוניות בעת שימוש ב- APM. אם לדוגמה - מדובר בפעולת I/O, אין צורך ב- Thread שימתין לתוצאה כלל ! הבקשה ממתינה בתור הביצוע של מנהל ההתקן, וכשהפעולה מסתיימת – מערכת ההפעלה קוראת לפונקציית הסיום שסיפקתם.
- (יותר נכון היא מוסיפה Work Item ל- Thread Pool של CLR, אבל זה כבר למי שמתעניין). זהו אחד הנושאים המוצגים בספר :

Jeffrey Richter (2006) – "CLR Via C#, Second Edition" (606)

1.7.2 Service Oriented Architecture

Service Oriented Architecture, או בשמה המקוצר SOA, היא למעשה פרדיגמת פיתוח אשר הרעיון המרכזי בה מנסה משקף מודולריות אבסולוטית - לאחד פונקציונליות אפליקטיבית, ולחשוף אותה בתור "שירות" - עם תלות מינימלית ככל האפשר בצרכן.

הרעיון העומד מאחורי SOA למעשה נובע מצורך שהלך וגדל עם השנים לאינטראופרביליות עם מערכות אחרות שבסביבה - שכן כיום מערכות מודרניות אינן מונוליטיות ועליהן להשתלב באופן חלק ככל האפשר עם מערכות אחרות. תחת צורך זה - עולה המוטיבציה לפתח את המערכת המודרנית עם מאפיינים של יכולת שימוש חוזר (Reusability) ואינטראופרביליות.

כאשר אומרים "שירות", המשמעות הינה קונספטואלית – באותה מידה זה יכול להיות Web Service או שרת אשר מממש תקשורת באמצעות פרוטוקול ייחודי על גבי TCP/IP, אין זה משנה כהוא זה את הרעיון.

המטרות העיקריות של פרדיגמה זו הינן:

1. Reuse and Composition - היכולת לחשוף ולהשתמש ברכיבי מערכת שונים, בין מערכות שונות.
 2. Permanence - היכולת לתמוך בכניסה של מערכות וטכנולוגיות חדשות. (חיי מוצר ארוכים)
 3. Flexibility - גמישות – היכולת להוסיף רכיבי מערכת שיענו על צרכים עסקיים נוספים.
 4. Openness and Interoperability - על מנת שנוכל לשתף רכיבים בין אפליקציות ובין פלטפורמות שונות.
 5. Distribution - כך שרכיבי המערכת יהיו נגישים גם ממיקום מרוחק (שימוש בתקשורת).
 6. Performance - ביצועים, כמובן, בדגש על Scalability - היכולת לענות על צרכים בסדר גודל עולה, באמצעות תוספות יחסית פשוטות של שרתים/רכיבים תומכים, ללא צורך לבצע שינויים בארכיטקטורת המערכת.
- היתרון המשמעותי ביותר בשיטה, מנקודת מבט של מתכנת או מהנדס מערכת, הינו הרעיון של – Loose Coupling יצירת התלות המינימלית ביותר בין רכיבי מערכת אחד למשנהו. ככל שתלות זו קטנה יותר - כך רכיבי המערכת ניתנים לניהול באופן מרוכז יותר :
- שינויים בקוד המקור של המערכת הופכים לפשוטים ומיידיים יותר
 - הוספת תמיכה לצריכה מוגברת של שירותי שכבה מסויימת הופכת לקלה יותר (Scale-up)
 - ניתן להחליף לחלוטין שכבה מסויימת בשכבה אחרת אשר חושפת ממשק חיצוני זהה (לדוגמה DAL - אשר נכתב ל MS-SQL - אבל עלינו לעבור ל - Oracle - ניתן לכתוב את ה- DAL בלבד ולהחליף בקיים, מבלי שזה יגרור שינויים בשאר המערכת. כמובן, באופן תיאורטי ☺).

ניתן להרחיב את ההבנה בנושא באמצעות המקורות הבאים:

Thomas Erl (2006) - "Service-Oriented Architecture (SOA): Concepts, Technology, and Design"

Fabrice Marguerie (1994) – "Getting a little closer to SOA"

- <http://madgeek.com/Articles/SOA/EN/SOA-Softly.html>,

1.7.3. השימוש ב-DLinq כתשתית ORM

ככל שהתפתח עולם התוכנה, כך הדרישות למבני מידע מורכבים, הן בקוד והן במסד הנתונים, גברו. בצד הקוד - הנהירה ל- Object Oriented, ובצד מסד הנתונים – הרחבת יכולות התשאול, סוגי שדות חדשים וכלים שונים אשר מקלים על ניהול המידע הרב הנצרך כיום על ידי אפליקציות מודרניות.

אך לאורך קו החיים של שני הצדדים - התוכנה ומסד הנתונים - קיימים, כידוע, פערים באופן ייצוג המידע.

מחד, מסד הנתונים אינו מאפשר ייצוג ישיר של מנגנוני ירושה, פולימורפיזם, הכלה, כנהוג ב- O.O Programming -ומאידך - המנגנונים והכלים הסינטקטיים הקיימים בשפות מונחות עצמים אינן עשירות ביכולות תשאול המזכירות את אלו של מסד הנתונים.

פערים אלו גורמים לכך ש-Persistence (שמירת מידע, ויכולת אחזורו מאתר אחסון) של ישויות אפליקטיביות הפך למשימה המצריכה מאמצי פיתוח משמעותיים (שורות קוד רבות ומייגעות).

פערים אלו ונוספים, הובילו את מודלי התוכנה הנפוצים להכיל שכבת תוכנה שלמה, המיועדת לניהול ההתקשרות של האפליקציה מול מסד הנתונים – בשמה המקובל - ה- DAL (Data Access Layer).

על כך נוספת העובדה שלמעשה מדובר במשימת "תרגום" מייצוד אחד לאחר – מה שלמעשה מגביר את המוטיבציה לאוטומציה של התהליך, על מנת שמאמצי הפיתוח יוכלו להתרכז ב-Core Logic של האפליקציה.

המתודולוגיה הנפוצה לפתרון בעייה זו נקראת Object Relational Mapping - ORM. הרעיון מאחורי מתודולוגיה זו הינו קיומה של תשתית תוכנה כללית, המאפשרת מיפוי של המודל מונחה העצמים למודל הרלציוני של מסד הנתונים, ובכך – לחסוך מאמצי פיתוח ניכרים.

תשתית ORM מאפשרת לנו לממש Persistence בלי לכתוב שורת SQL בודדת. הוא מובנה בצורה המממשת אידיומות של OOP (Object Oriented Programming) ומאפשר לנו לשמור, לשלוף ולתשאל מבני נתונים, אפילו מורכבים - ולקבל בחזרה את התוצאות כאובייקטים.

אחת התשתיות הנפוצות כיום בעולם ה-Java, ואולי המפורסמת ביותר באופן כללי, הינה תשתית הנקראת Hibernate וכמו במקרים נוספים - עולם ה-Net. לא יכל להישאר מאחור - ונפתח פרוייקט NHibernate, אשר כיום מהווה תשתית ה-ORM הנבחרת ברוב הפרוייקטים במערכות מידע.

אמנם באיחור מה, אך עם הבטחה גדולה – Microsoft החליטה להוציא, יחד עם .Net 3.5 את DLinq – אשר בין השאר – מהווה פתרון DAL אשר מתחרה בכל אלו, עם יתרונות וחסרונות מסויימים.

אחד היתרונות הבולטים בטכנולוגיה זו הוא היכולת לבנות שאילתה אל מול מסד הנתונים כחלק מהקוד – ובכך לקבל את הכל היתרונות של תמיכת ה-Compiler של השפה – Type Safety, Error Detection וכו'.

אני לא אכנס בסקר הספרות לפרטים הטכניים - שכן עקומת הלימוד של שימוש (ברמה בסיסית) ביכולות של תשתית זו מוערך לרוב בכשבעים – מפאת הפרטים והיכולות הרבות.

אתר המקור –

<http://msdn.microsoft.com/en-us/netframework/aa904594.aspx>

1.8. תאור מצב קיים כולל ניתוח חלופות מערכת

1.8.1. ניתוח חלופות מערכת

כיום קיימות מספר מערכות דומות, כגון DeviantArt.com העולמית, ו-stage.co.il או cgart.co.il הישראליות.

a. DeviantArt.com – יש שיגידו "המלכה הבלתי מעורערת" של התחום, אחד האתרים הפופולריים ביותר אשר חולש על כל קטגוריה אפשרית – מצילום, דרך סרטונים, עיצוב תעשייתי, Comics, Anime ועוד רבים. מוכר יחסית בישראל, וניתן למצוא בו עבודות איכותיות, אך ברזולוציה נמוכה יחסית. מהות האתר היא העיקר הקהילה שנוצרה סביבו, ושאיבת השראה אחד מעבודתו של השני. כמו כן, האתר מציע למכירה הדפסים של התמונות הפופולריות באתר, ומתגמל את האומנים על רכישות של יצירותיהם.

האתר מעוצב היטב והינו נוח לשימוש, אך הוא אינו ממוקד דיו בעיצוב חזותי בקהל יעד מקצועי/סטודנטיאלי, וגם אינו מספק מספיק כלים לקהילה "להבליט" ולתמוך בעבודות היוצרים מעבר למתן תגובות.

b. cgtalk.co.il – האתר הינו יחסית מוכר בקרב אנשי התעשייה והסטודנטים הישראליים, אך ממשק המשתמש בו אינו עולה בקנה מידה אחד עם ממשקים מודרניים, והוא מזכיר יותר מערכת פורומים ואינו מספק מערכות דירוג. במבט ראשוני ניתן לראות כי ישנה פעילות של העלאת יצירות, אך לא קיימת פעילות קהילתית כמעט כלל – מספר ההערות מועט, אם בכלל, אין דירוגים, אך ישנם קישורים לתחרויות ואירועים ישראליים בתחום.

1.8.2. ניתוח חלופות מערכת

נכון להיום לא נצפתה מערכת זהה או דומה בסדר גודל המצדיק היותה חלופה מערכתית. DeviantArt.com, אשר הכי קרובה למערכת המוצעת במהותה, הייתה יכולה לשמש חלופה מערכתית באופן משביע רצון אילו היה דגש על תחום העיצוב החזותי באתר, ודגש כבד יותר במיקוד קהל היעד המקצועי/אקדמי מאשר בספקטרום רחב יותר, כפי שקיים היום.

- שפה – אנגלית בלבד.
- מיקוד – תחום האומנות הויזואלית באופן כללי, כולל צילום, אנימציה, תמונות רקע למסך המחשב וכדומה. יותר "מקום שיש בו תמונות יפות", מאשר קהילה.
- קלות שימוש – המערכת הינה קלה לשימוש וידידותית.
- ביצועים – רמת הביצועים הינה טובה (שניה עד שניה וחצי לעמוד, ללא תמונות).

טבלה 1

deviantArt.com	porT	
5	5	ידידותיות למשתמש
4	5	מיקוד לתחום אומנות דיגיטלית
3	5	מיקוד לקהל מקצועי/ אקדמאי
4	4	קלות שימוש
5	4	ביצועים
0	5	תמיכה בעברית

בניתוח המערכתי המשקל של כל אחד מהסעיפים זהה, וניתן לשקלל את ציון החלופות :

porT – 93%

deviantArt.com – 70%

2. דרישות המערכת (Software Requirements)

- יצירת חשבון אישי – אימייל וסיסמה
- איחסון והצגה של תמונות לפי קטגוריות
- שמירת "הערות היוצר" על היצירה
- הצגה של תמונות בפורמטים שונים ובגדלים שונים
- יצירת תיק עבודות אישי ובהצגה פומבית של תיק עבודות זה
- דירוג יצירות
- כתיבת Comments על יצירות
- פורום
- סימון "יצירות אהובות" לתוך תיקיית "מועדפים" אישית

השינויים העיקריים מהצעת הפרוייקט :

- הוספה תמיכה בסינון מתקדם של יצירות – לפי זמנים, קטגוריות וקטגוריות אב
- הוספה תמיכה בתפריטים קונטקסטואליים
- תמיכה ב- Net Cardspace. - הטכנולוגיה לא התפתחה כפי שהיה צפוי, והפופולריות שלה עדיין נמוכה. תמיכה בטכנולוגיה זו תישקל שוב בעתיד.
- "יצירה בהמשכים" - תמיכה ברכיב זה ירדה, עקב שיחות עם יוצרים צעירים לגבי הצורך בו, לעומת כמות ההשקעה שנדרשה למימוש.

חוסרים ורכיבים שטרם הושלמו :

- דירוג של יצירה מושפע באופן שונה ממי שיש לו הרבה נקודות וממי שיש לו פחות
- ניהול החשבון האישי

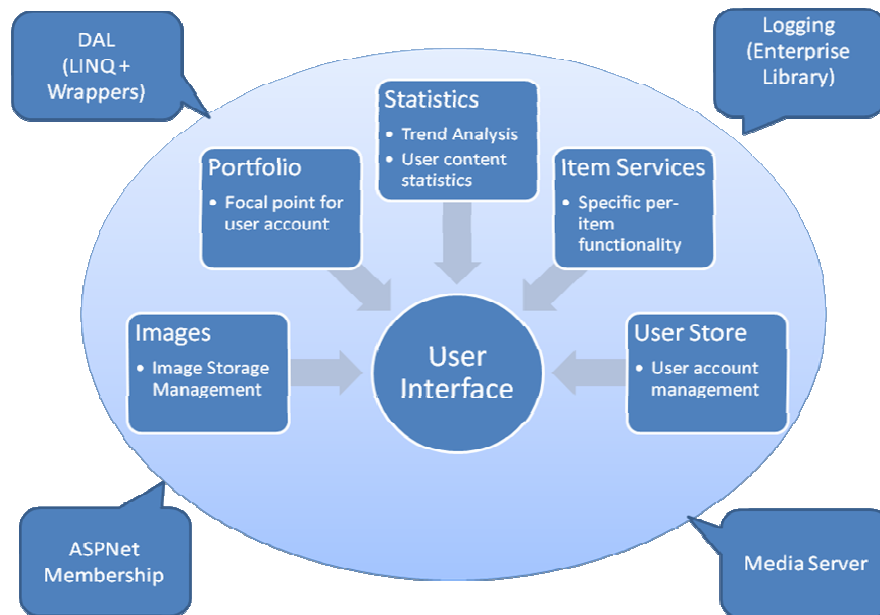
הרכיבים שטרם הושלמו הוסרו לעת עתה על מנת לאפשר פיתוח של רכיבים אחרים שחשיבותם צפה במהלך הפיתוח.

ניתן לבחון דרישות אלו אל מול דרישות המערכת המקוריות שהוצגו בהצעת המערכת, המופיעה כנספח שישי למסמך זה, בסעיף 3.

3. אפיון המערכת (Software Specifications)

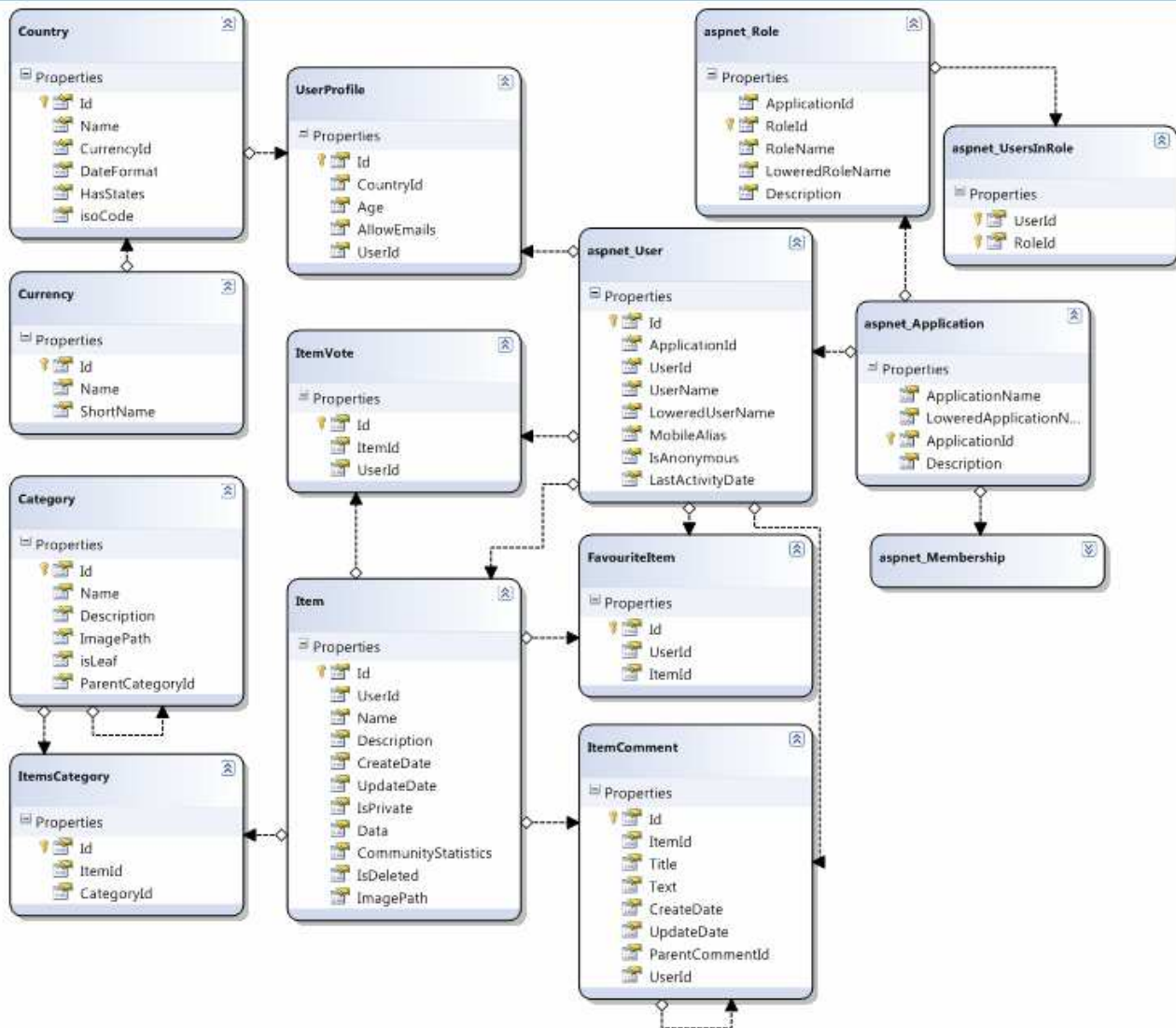
3.1 מודל המערכת

תרשים המערכת הבא מייצג את רכיבי המערכת העיקריים, ואת חלוקתם לשכבות. ממשק המשתמש במרכז, נותני השירות העיקריים שלו במעגל הפנימי, נותני שירות משניים, ורכיבי תשתית נוספים – במעגל החיצוני :



איור 1 – המערכת במבט על

ERD של המערכת:



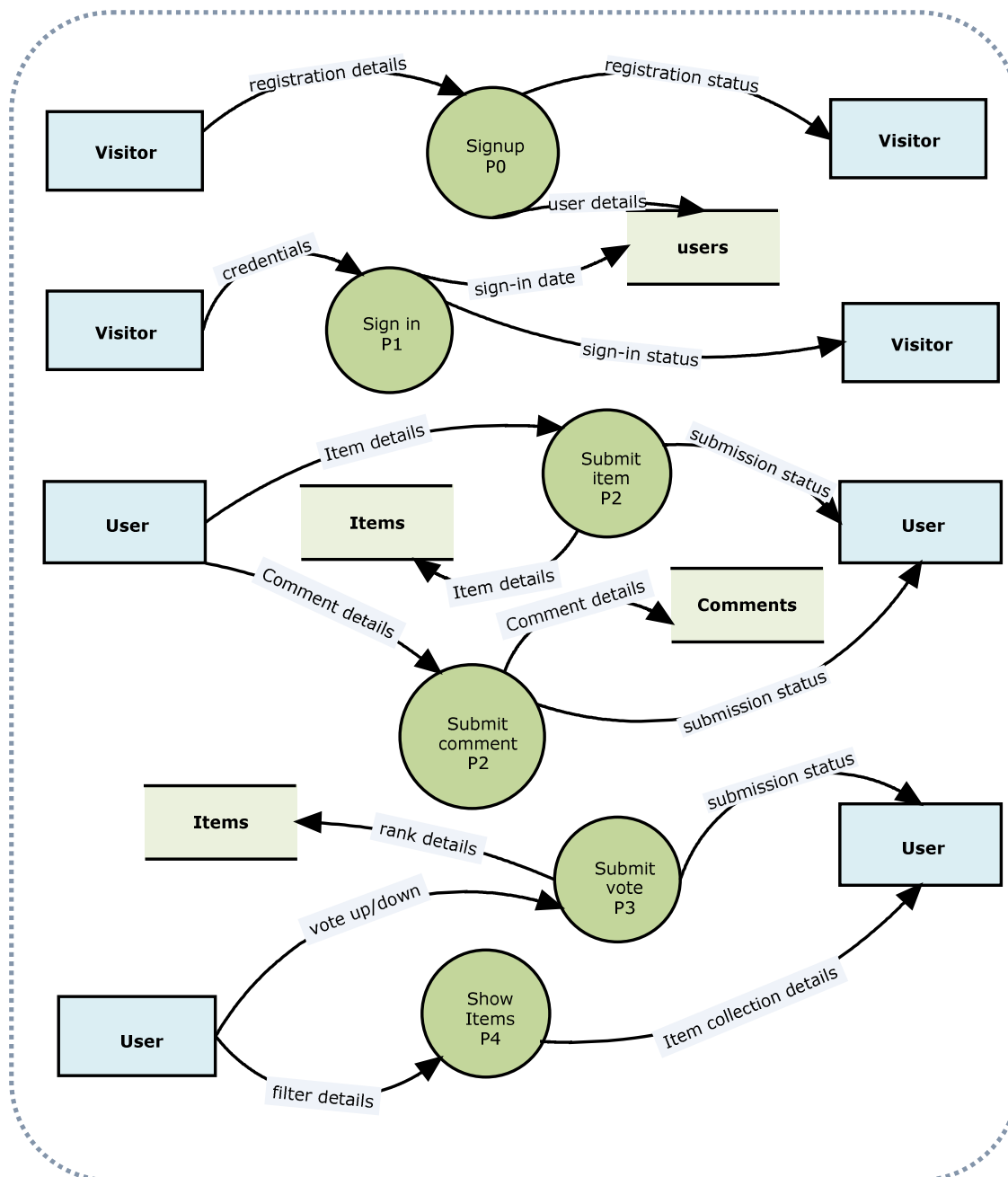
איור 2 - ERD

פירוט נוסף של מודל המערכת קיים בנספח SRD של מסמך זה – סעיף 2.7.

3.2. אפיון פונקציונלי

להלן DFD-0 של עיקרי פעולות המשתמש במערכת :

DFD 0 - porT



איור 3 - DFD

פירוט מלא של כל האפיון הפונקציונלי ניתן למצוא בסעיף 3.1 של מסמך ה-SRD, המצורף כנספח למסמך זה.

3.3. ביצועים עיקריים

דרישות הביצועים העיקריות נגזרות מאופי המערכת – ממשק משתמש המיועד לקהל רחב של משתמשים, לא יכול להסבול טעינה איטית מידי של מידע או ביצוע פעולת משתמש בזמן שאינו סביר.

ישנם פרמטרים אשר אינם ניתנים להשפעה כגון רוחב פס של חיבור המשתמש, עומס, רוחב פס של חיבור בין יבשתי וכדומה. הפרמטרים המוצגים כאן מתוארים לגבי בדיקה על מול מערכת שאינה בעומס, ובחיבור מקומי.

- על עמודי המערכת להיטען תוך זמן מקסימלי של 2 שניות מרגע קבלת הבקשה.
- פעולות בממשק המשתמש המצריכות טעינה חלקית בלבד של העמוד – שימוש ב-AJAX
- מספר שאילתות מקסימלי למסד הנתונים במהלך טעינת עמוד - 20.
- על השדות הרלוונטיים לשאילתות להיות מאונדקסים כנדרש במסד הנתונים.
- זמן שאילתה מקסימלי – 700ms.

3.4. אילוצי סביבה

המערכת הינה מערכת WEB ועל כן עליה להיות נגישה מכל מקום ברשת האינטרנט. המערכת תרוץ תחת שרת IIS, על שרת מסוג Windows Server 2003.

המערכת תשב באותה סביבת LAN בו נמצא מסד הנתונים, מסוג Sql Server 2005.

פירוט נוסף ניתן למצוא בסעיף 2.4 של נספח ה-SRD, בסוף מסמך זה.

3.5. ניתוח חלופות טכנולוגיות

נחלק את המערכת לחלקים השונים, וננתח את החלופות הטכנולוגיות -

3.5.1. טכנולוגיית פיתוח

טכנולוגיית הפיתוח שנבחרה היא ASP.Net, ופיתוח בשפת C#. החלופות האפשריות הינן רבות, נבחן מספר מהפופולריות שבהן –

- ASP – טכנולוגיית פיתוח Web המבוססת על vbscript. כיום איבדה פופולריות אך עדיין חזקה מאוד ומיוצגת בגאווה על ידי אתרים כמו zap.co.il ועוד רבים. טכנולוגיית פיתוח זו לוקה במספר תחומים - ספריות בסיס דלות גורמות לקוד מסורבל ולא קריא ולהשקעת זמן פיתוח במימוש חלקים אשר קיימים מראש בטכנולוגיות אחרות, וקושי בהפרדה בין התוכן המוצג לבין לוגיקת הליבה של המערכת.

כמו כן – קשה למצוא רכיבי הרחבה בקוד פתוח משמעותיים לטכנולוגיה זו, בעיקר בתחומים הקשורים לממשק מול מסד הנתונים.

- PHP – פחות פופולרית בארץ לעומת שאר אירופה, בה השימוש בה נפוץ מאוד. טכנולוגיה מבוססת אשר התפתחה מאוד בשנים האחרונות. מלווה ומפותחת על ידי חברת Zend הישראלית. דומה מעט ב-Syntax וב"סגנון הפיתוח" ל-ASP.
- Ruby – טכנולוגיה צעירה יחסית, אשר לוקחת את מודל ה-MVP (Model View Presenter) עד הסוף ומקלה על המפתח בתחומים רבים – מפתחים המשתמשים בטכנולוגיה זו הינם גם התומכים הנלהבים ביותר שלה, שכן סגנון הפיתוח הייחודי מתמקד בפשטות ופרודוקטיביות מעל לכל, שכן הטכנולוגיה מספקת אוטומציה בתחומים רבים, העיקרי שבהם – הגישה למסד הנתונים, ובניית מודל הנתונים.

Ruby	PHP	ASP	ASP.Net	
3	3	3	4	קלות פיתוח
1	2	4	5	היכרות המפתח עם הטכנולוגיה
D	D	D	C	Dynamic / Compiled
4	5	4	3	ביצועים
4	5	1	1	תמיכה בפלטפורמות מרובות
1	4	3	4	קהילה, תיעוד ותמיכה

בשקלול הנתונים (משקל זהה לכל הסעיפים – 10%, מלבד היכרות המפתח – 50%)

3.7/5 – ASP.Net

3.1/5 – ASP

2.7/5 – PHP

1.7/5 – Ruby

3.5.2. מסד נתונים

- Oracle – שרת מסדות נתונים מבוסס ווותיק, אשר ידוע ביתרונו המשמעותי בתחום ה-Enterprise – תמיכה במספר רב של משתמשים, יכולות עיבוד גבוהות, ויעילות. יחד עם כל אלו מגיע תג מחיר גבוה יחסית לעסקים אשר רוצים לרכוש שרת מבוסס Oracle.
- MySQL – מסד נתונים נפוץ מבוסס קוד פתוח, אשר מוכר בעיקר בגירסת ה-Community המופצת בחינם. זהו שרת ותיק אשר בדרך כלל מהווה את ה"חלק האחורי" של אפליקציות PHP ו-Ruby, הן משום הקלות בפיתוח והן משום ה"שייכות" שהתפתחה לשרת זה עם עולם הלינוקס/Unix.

MySql	Oracle/DB2	SQL Server 2005	
0	0	5	תמיכה במנגנוני Dependency Cache של ASP.Net
2	1	5	היכרות המפתח עם המערכת
4	4	4	נוחות תחזוקה ושינוי הגדרות
4	5	3	ביצועים
x	x	v	אינטגרציה קלה עם Linq
v	v	x	תמיכה בפלטפורמות מרובות
x	x	v	תמיכה ב- CLR Integration

בשקלול הנתונים (משקל זהה לכל הסעיפים – 10%, מלבד היכרות המפתח – 50%)

3.7/5 - SQL Server 2005

1.4/5 – Oracle

1.8/5 - MySQL

3.5.3 שרת Web

- Apache – שרת Web המשמש בדרך כלל לשרתי Unix/Linux לחשיפת אפליקציות PHP, אך קיים גם בגירסת Windows. אין לשרת זה תמיכה (כיום) בתשתית ה-.Net, אך ישנו פרוייקט קוד פתוח בשם Mono, אשר מנסה לגשר על פער זה.

Apache	IIS	
0	5	תמיכה ב- ASP.Net
2	5	היכרות המפתח עם הטכנולוגיה
2	4	נוחות תחזוקה ושינוי הגדרות
4	3	ביצועים
5	0	תמיכה בפלטפורמות מרובות
0	5	קוד מנוהל

בשקלול הנתונים (משקל זהה לכל הסעיפים – 10%, מלבד היכרות המפתח – 50%)

4.2/5 - IIS

2.3/5 – Apache

3.5.4. הפתרון שנבחר

טכנולוגיית פיתוח – ASP.Net – נבחרה בעיקר משום שיש למפתח המערכת נסיון קודם בפיתוח בטכנולוגיה זו. יתרונותיה של טכנולוגיה זו באים לידי ביטוי במודל פיתוח מובנה ומבוסס, פיתוח מונחה אובייקטים, והיכולת להשתמש ברכיבים ובמחלקות רבות אשר קיימות כחלק מתשתית ה-Net.

יתרונות עיקריים :

1. מיומנות קיימת של המפתח בטכנולוגיה.
2. יכולות Object Oriented והתאמה למתודולוגיית פיתוח Test Driven, באמצעות כלים כגון NUnit.
3. יכולות Ajax (חויית משתמש משופרת) קלות להטמעה באמצעות שימוש במוצרי תשתית קיימים – חיסכון בזמן פיתוח.
4. יכולות ORM באמצעות NHibernate.

שרת Web – IIS – כיום שרת ה-WEB היחיד אשר מסוגל לשרת תוכן של

ASP.Net, ולכן נבחר.

מסד נתונים – SQL Server 2005 – מסד נתונים זה נבחר עקב נסיון קודם שיש למפתח המערכת עם מסד נתונים זה, ובנוסף – עקב פשטות הפיתוח היחסית, והתאימות לתשתית – NHibernate.

יתרונות עיקריים של מסד נתונים זה –

1. מיומנות קיימת של המפתח בעבודה מול מסד נתונים זה.
 2. רמת תאימות גבוהה ל- NHibernate
 3. מאפשר הטמעה קלה ושימוש ב- Cache Dependency של Net, באמצעות Polling או SQL Server Agent. (מנגנונים תומכים עבור in-memory caching).
- על מנת להרחיב אודות דרישות המערכת בהתאם לפיתרון שנבחר, יש לפנות למסמך ה-SRD של הפרוייקט.

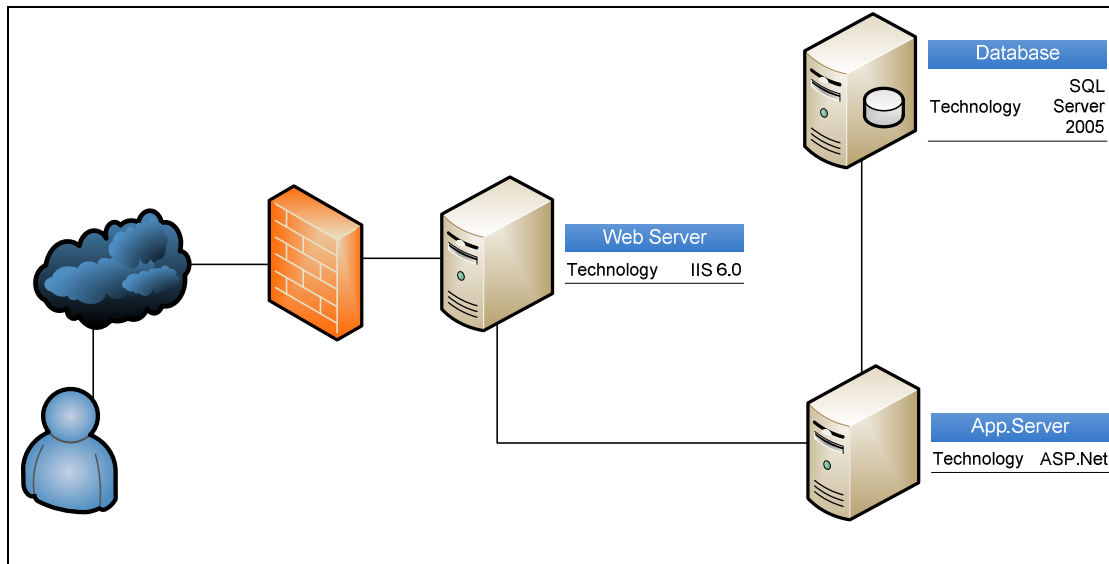
4. תיכון המערכת

4.1. ארכיטקטורת המערכת

תיכון המערכת יתבצע על סמך מתודולוגיית OOD, כאשר ישנה חלוקת אחריות למספר שכבות פונקציונליות :

- User Interface
- Service Layer
- Entity Layer
- Data Layer

תיאור כללי של ארכיטקטורת המערכת :

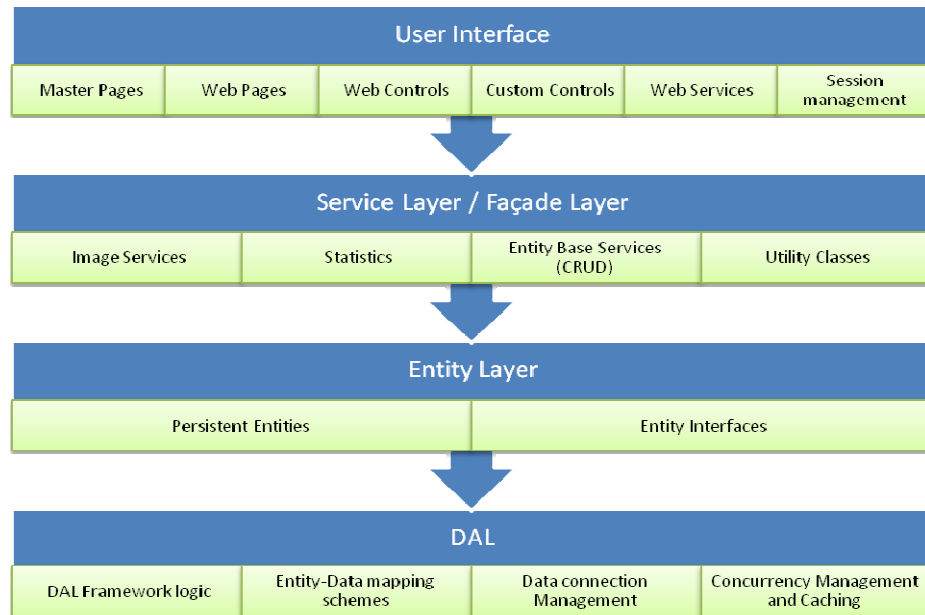


איור 4 - ארכיטקטורה

ניתן לבחון את חלוקת האחריות והפונקציונליות בכל שכבה, וכמו כן להעמיק לפרטי תתי הרכיבים של המערכת בנספח ה-SDD, בסעיף 2.

4.2. תיכון מפורט

להלן תרשים כללי של עיקרי הרכיבים של המערכת :

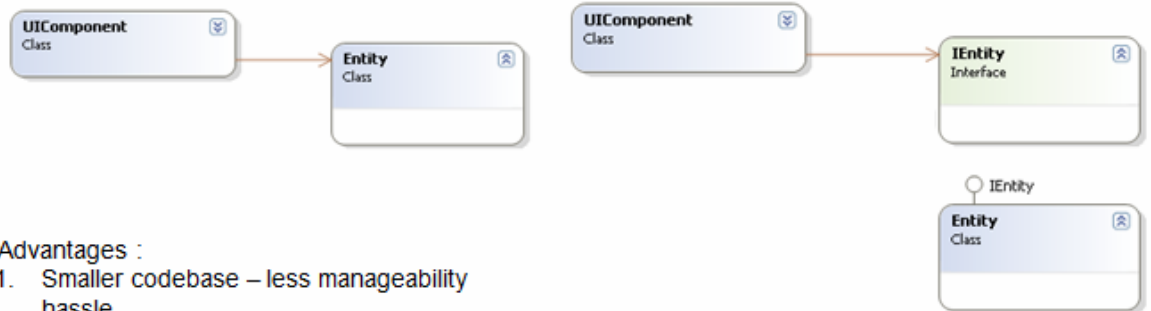


ניתן לבחון את חלוקת האחריות והפונקציונליות בכל שכבה, וכמו כן להעמיק לפרטי תתי הרכיבים של המערכת בנספח ה-SDD, בסעיף 3.

4.3. אלטרנטיבות לתיכון

עדיף להציג נושא זה באנגלית, על מנת לדייק בעולם המושגים :

1. Entities – using and maintaining interfaces, vs. not having entity interfaces.



Advantages :

1. Smaller codebase – less manageability hassle.
2. When entities don't contain their own functionality – removes redundancy of encapsulation.
3. Less confusion – always working with same entity abstraction level, regardless of layer.

Disadvantages :

1. Tightly coupled layers
2. Less encapsulation
3. Not future-ready (in some cases...)

Advantages :

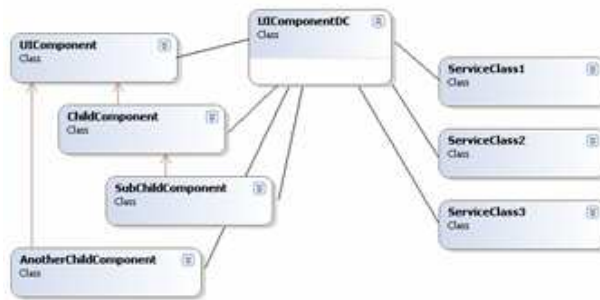
1. Separation of concerns – reducing the coupling between layers.
2. Preparation for future – scales up well, and forces you to encapsulate functionality.

Disadvantages :

1. Maintenance of interfaces on entity interface changes
2. Redundancy if not used

The selected method for this project is the one on the left, due to the fact that the benefits are greater for a project at this stage, and if there will come a later need – entities can always be refactored into interfaces.

2. UI Component Data Access – DataCart vs. Autonomous access

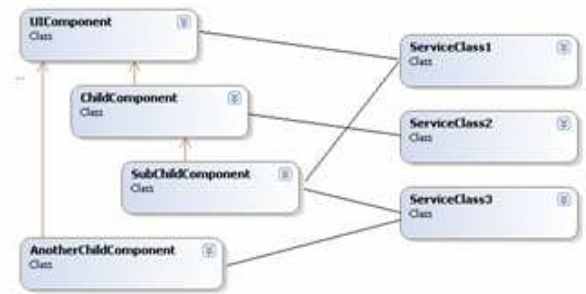


Advantages :

1. Forces pre-planning
2. Less clutter in maintenance - data retrieval is concentrated in one place

Disadvantages :

1. More code to manage
2. Single point of failure



Advantages :

1. More fluent implementation – less pre-planning.
2. Better in simple scenarios, and in scenarios where each UI component has a strict domain scope.

Disadvantages :

1. Service Class interface change causes more refactoring (reference duplication)
2. Duplicate service method calls – higher probability

The selected method for this project is the one on the right, due to the fact that the benefits are greater for a project at this stage, and if there will come a later need – a DataCart can always be refactored into from autonomous access.

3. Media Storage - File system? SQL Server?

SQL Server

1. Is great for management.
2. Everything is in one place, relationally connected to relevant entities.
3. Is pretty fast (when you index what you look for)

File System

1. Can be directly exposed for web consumption
2. Easier to see the picture itself (don't need special code just to see all the images)
3. Is fast too, but...
4. Complex management (keeping tabs on data, related to the database, but not IN the database can be a nightmare)

Solution is a MediaServer – Saves on file system, retrieves for web, manages on SQL. (and a handfull to code, but component is critical for an image based application.)

5. תכנון הפרוייקט

5.1. ניהול סיכונים

סעיף ניהול הסיכונים מתואר במסמך ה- SPMP של הפרוייקט (סעיף 2). נספח זה מכיל, בנוסף, את הערכת הסיכונים לפיתוח הפרוייקט, ומתאר את מצב ההתקדמות הנוכחי שלו. לא קיים שינוי בהערכת הסיכונים ממה שהוצג בהצעת הפרוייקט המוצגת בנספח השישי.

5.2. תוכנית עבודה

לוחות הזמנים המעודכנים מתוארים בנספח ה- SPMP של הפרוייקט (סעיף 4). נספח זה מכיל, בנוסף, את הערכת הסיכונים לפיתוח הפרוייקט, ומתאר את מצב ההתקדמות הנוכחי שלו. לא קיים שינוי בלוחות הזמנים ממה שהוצג בהצעת הפרוייקט.

6. בדיקות והערכה (Software Testing and Evaluation)

6.1. תוכנית בדיקות תוכנה

תוכנית הבדיקות מתוארת במסמך ה- STP של הפרוייקט, המצורף כנספח למסמך זה.

6.2. דוח בדיקות תוכנה

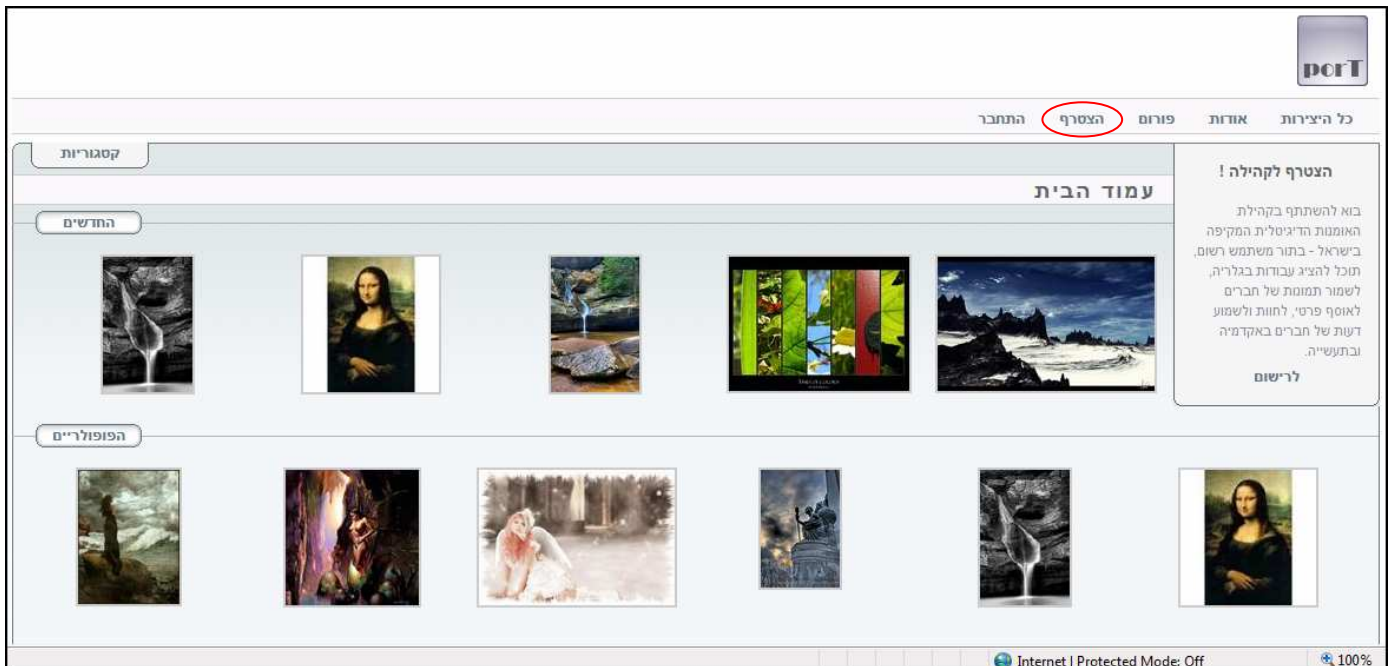
במהלך שלב הבדיקות נבדקו כל מרכיבי המערכת לחוד ויחדיו. בוצעו בדיקות debugger, unit test וכן בדיקות פונקציונאליות. ניתן לומר כי המערכת עברה את כל הבדיקות וכי היא עונה על דרישות המשתמש ומספקת את הפונקציונאליות הדרושה.

במסמך דוח בדיקות תוכנה (STR), ניספח רביעי בספר זה מופיעות הבדיקות שבוצעו לצורך בדיקת פונקציונאליות המערכת. המערכת עברה בהצלחה את כל הבדיקות המופיעות במסמך זה.

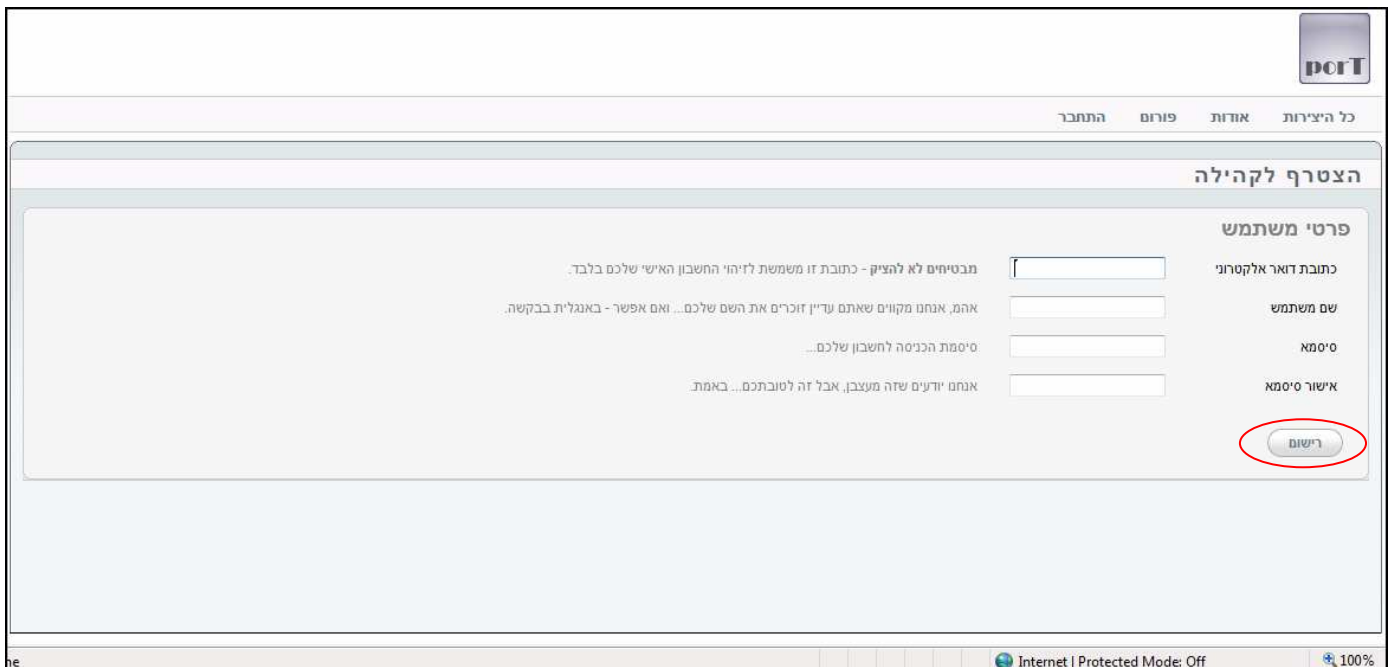
6.3. דוגמאות הפעלה מפורטות מקצה לקצה

6.3.1. רישום המשתמש למערכת

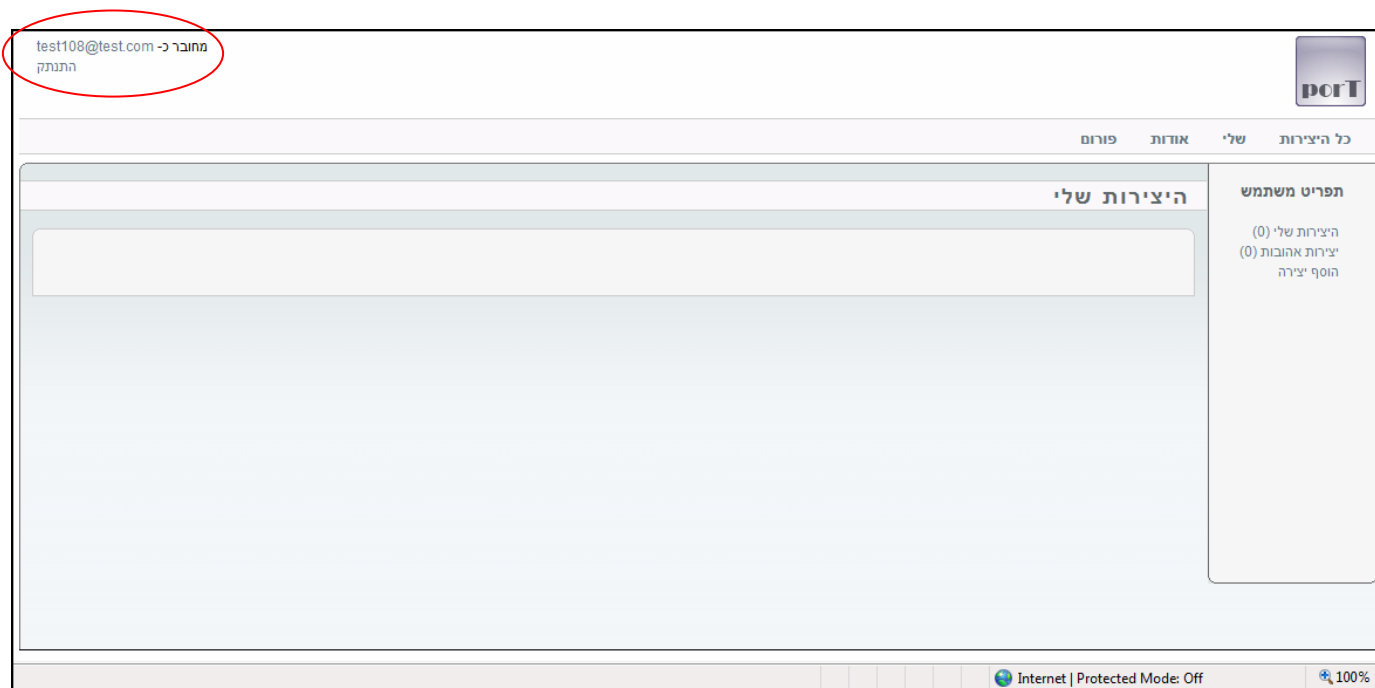
המשתמש מגיע לעמוד הראשי של המערכת, ומוצג בפניו עמוד הבית :



לאחר לחיצה על הקישור "הצטרף" יופיע בפניו המסך הבא :



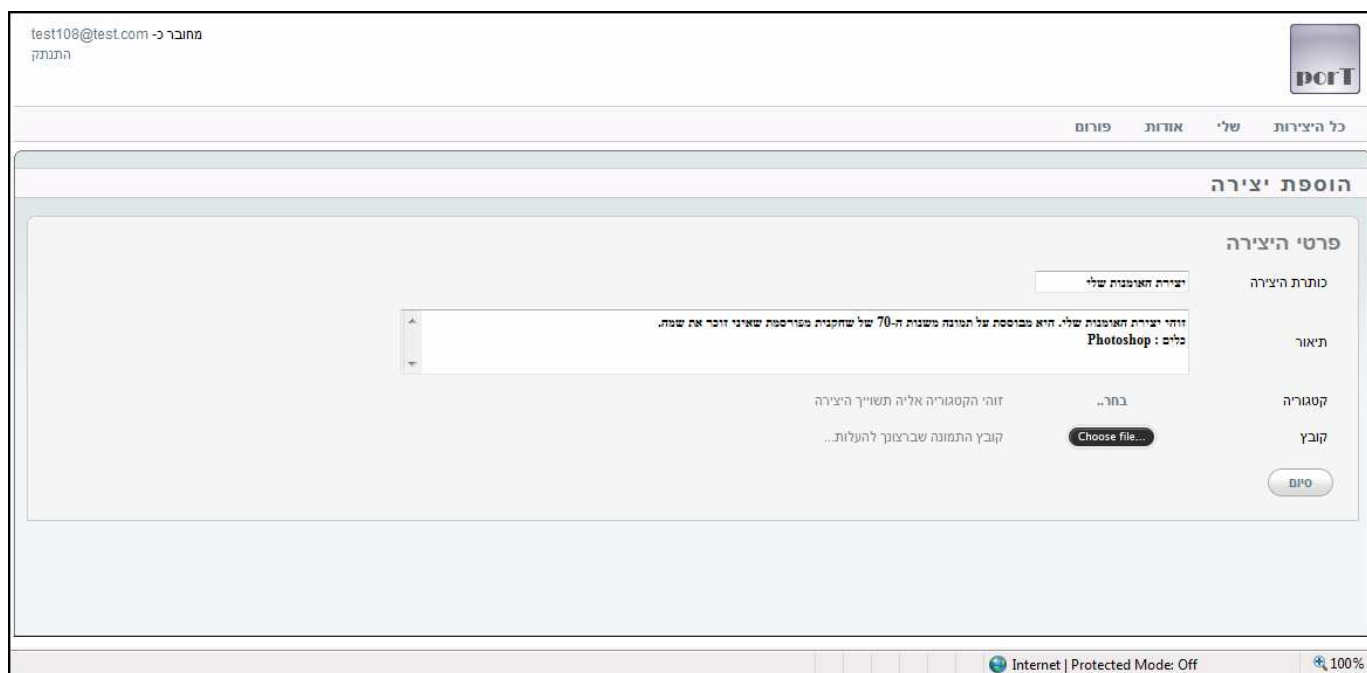
על המשתמש להקליד את פרטי הרישום שלו, ולאחר מכן ללחוץ על "רישום". הסיום התהליך – המשתמש יהיה מחובר למערכת, ויוצג בפניו המסך הבא :



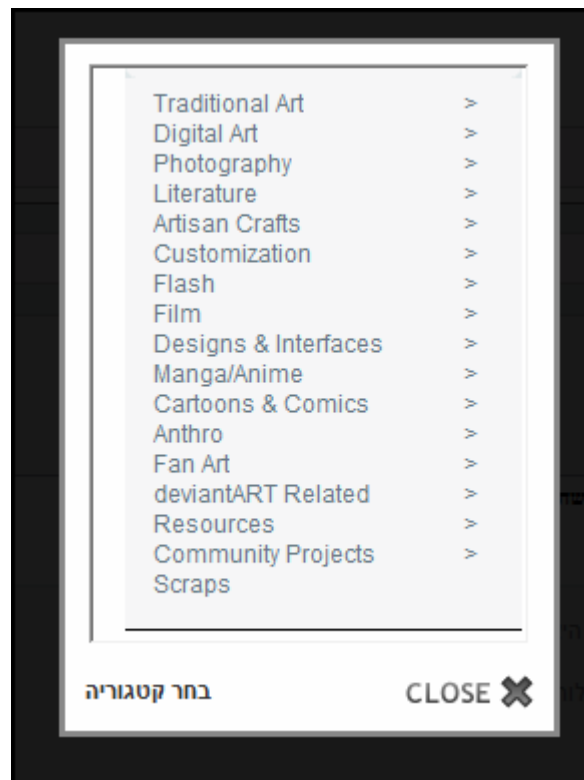
כעת המשתמש מחובר למערכת, ויכול להתחיל להוסיף יצירות.

6.3.2. הוספת יצירה

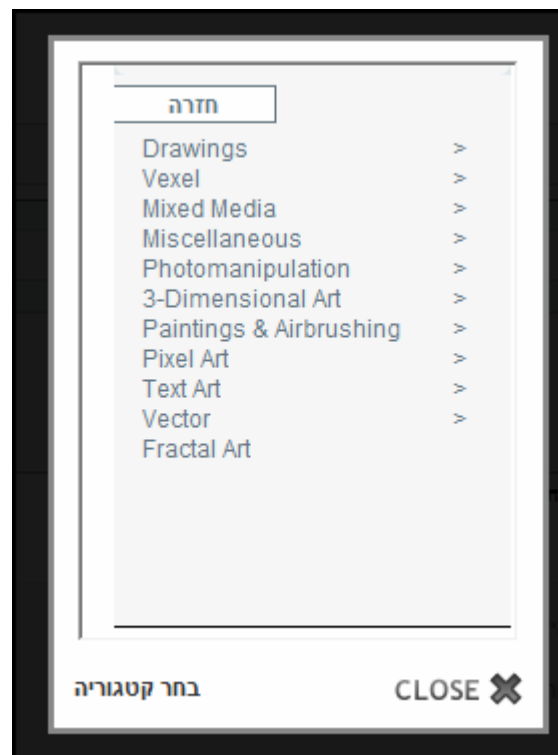
על מנת להוסיף יצירה, כאשר המשתמש מחובר עליו לבחור בקישור "הוסף יצירה" (ניתן לראותו במסך האחרון של תהליך הרישום, מצד ימין, בסוף תפריט המשתמש). יוצג בפני המשתמש המסך הבא :



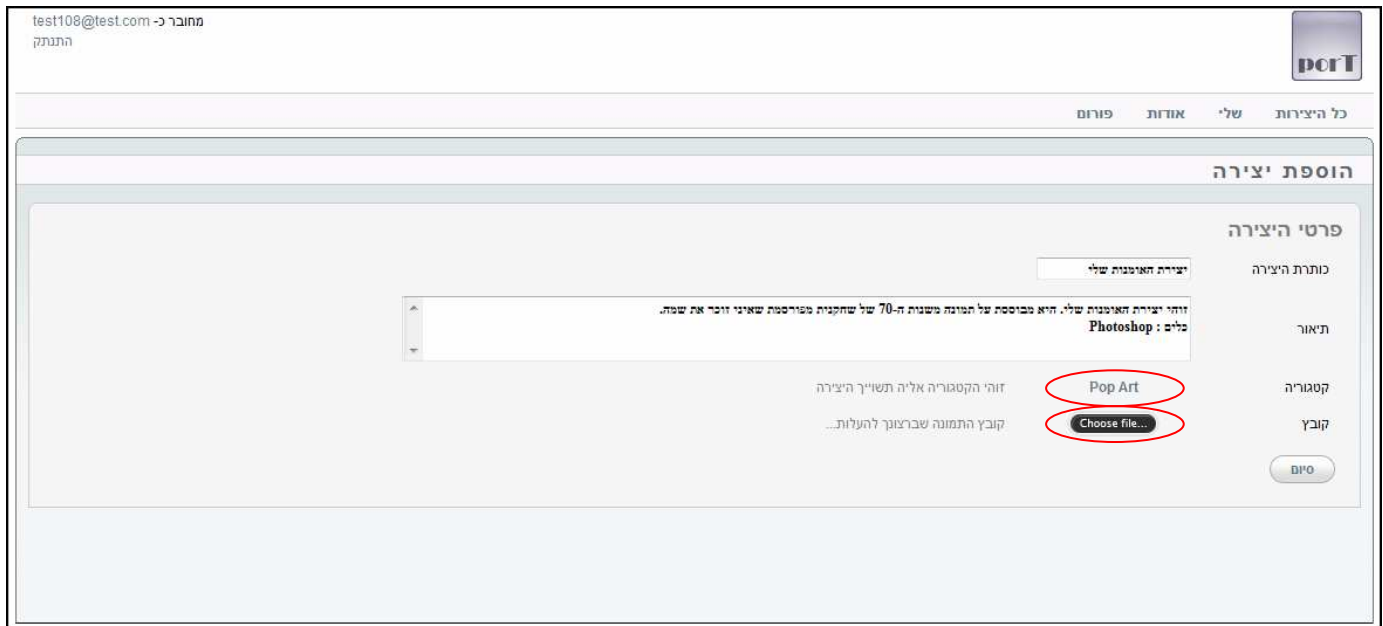
המשתמש ממלא את הפרטים, ולוחץ על מקש "בחר" על מנת לבחור קטגוריה. המסך מוחשך, ומוצג בפני המשתמש הממשק הבא :



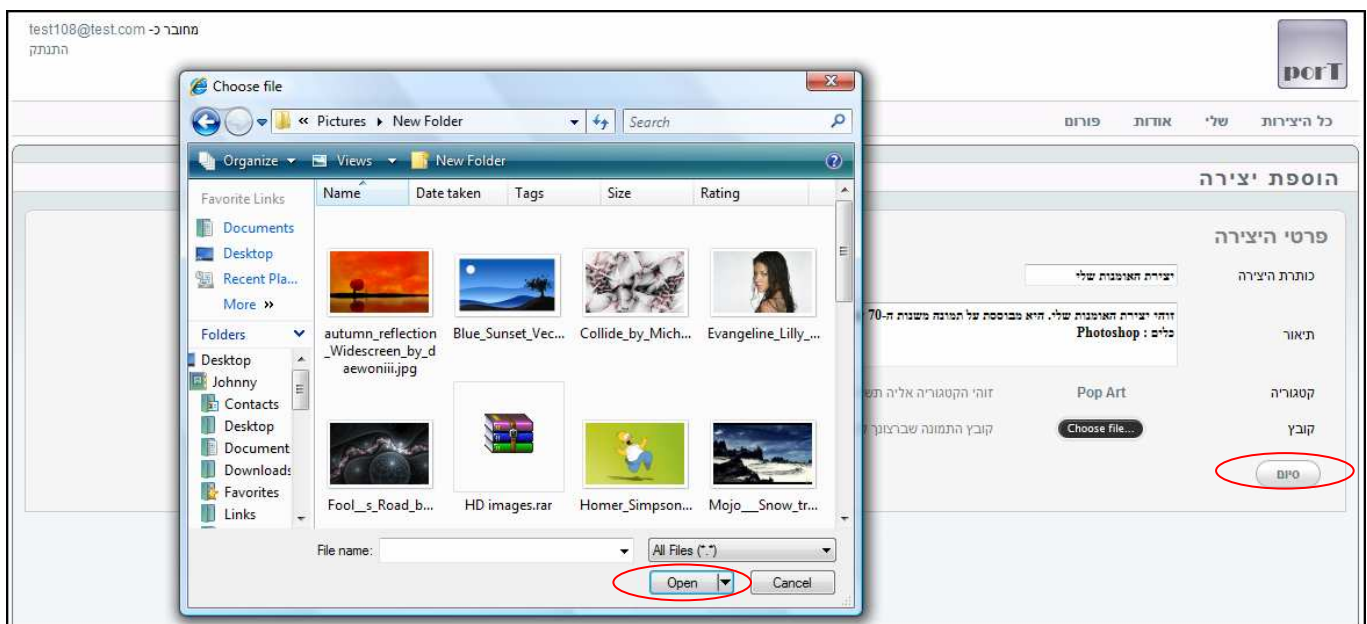
באמצעות ממשק זה המשתמש מנווט בין הקטגוריות, עד שהוא בוחר קטגוריה מסוימת. אם המשתמש לחץ לדוגמה על "Traditional Art", יוצגו כל תתי הקטגוריות הרלוונטיות :



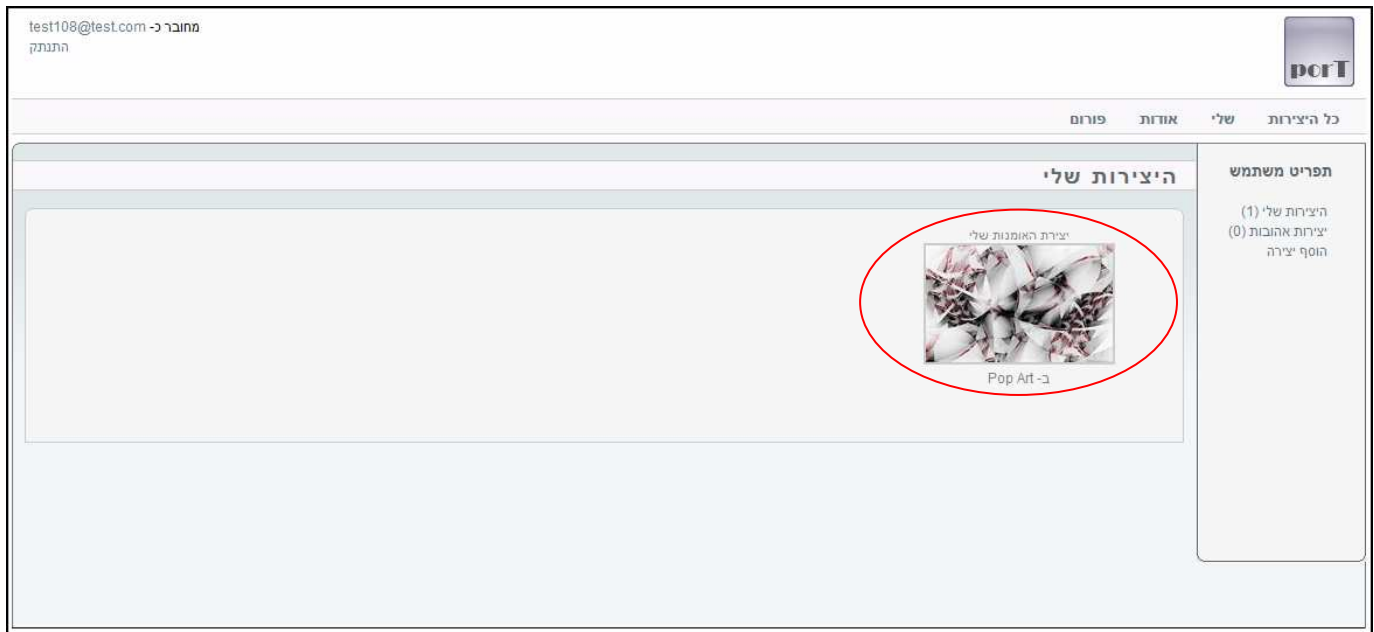
לבסוף, משנבחרה קטגוריה מסויימת, מוצג למשתמש שוב המסך העיקרי. הקטגוריה שסימן מוצגת, ועליו לבחור קובץ להעלאה :



המשתמש בוחר בכפתור "Choose File", ומוצג בפניו ממשק בחירת קובץ סטנדרטי של מערכת ההפעלה :

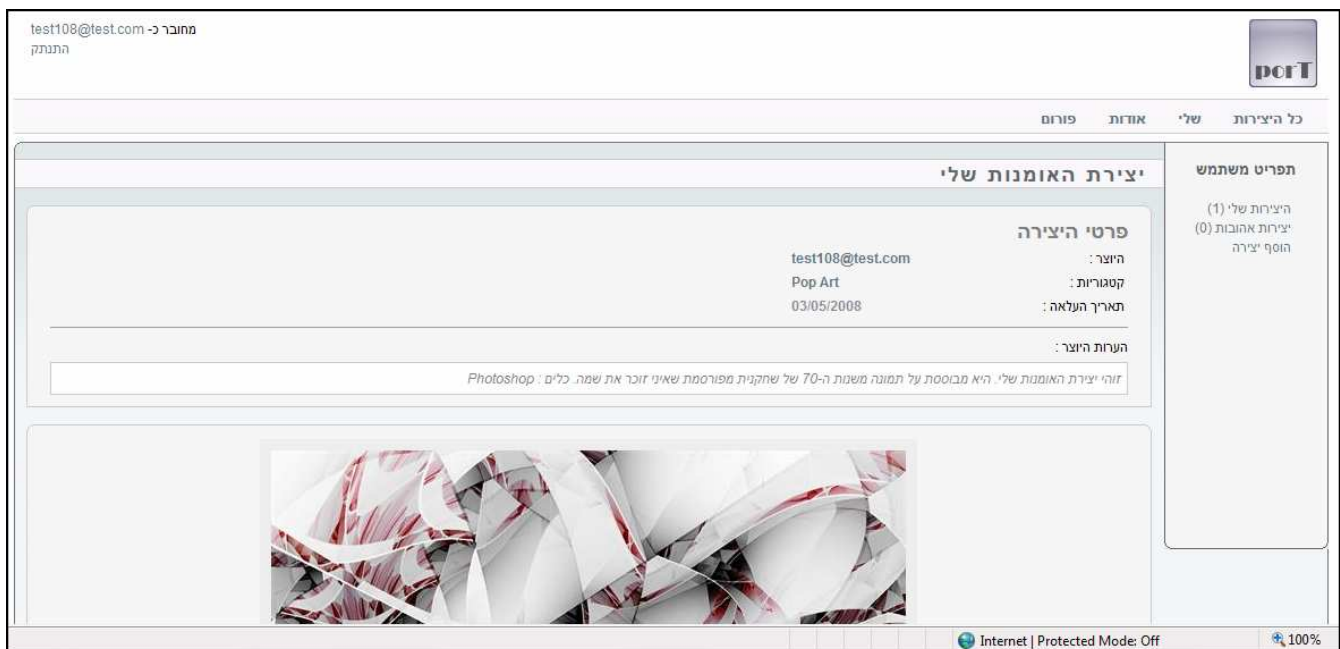


הוא בוחר את הקובץ הרצוי, ובוחר "סיום". כעת מוצג בפני המשתמש מסך "היצירות שלי":

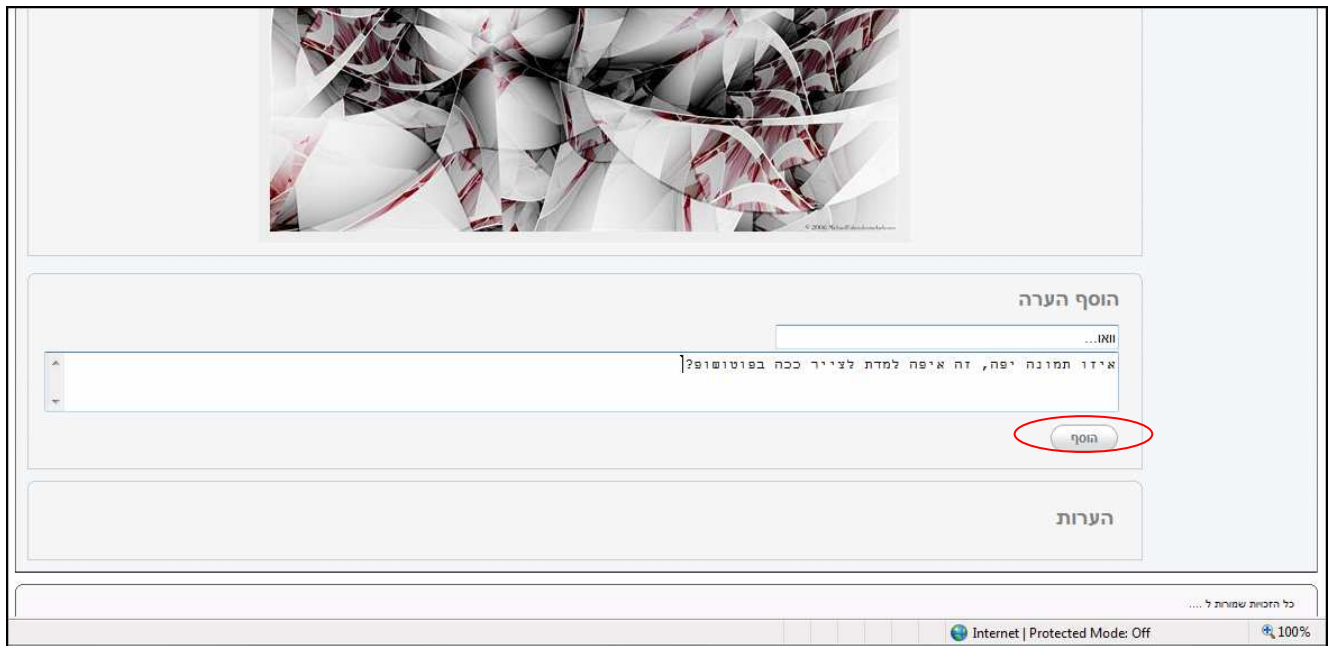


6.3.3. הוספת הערה

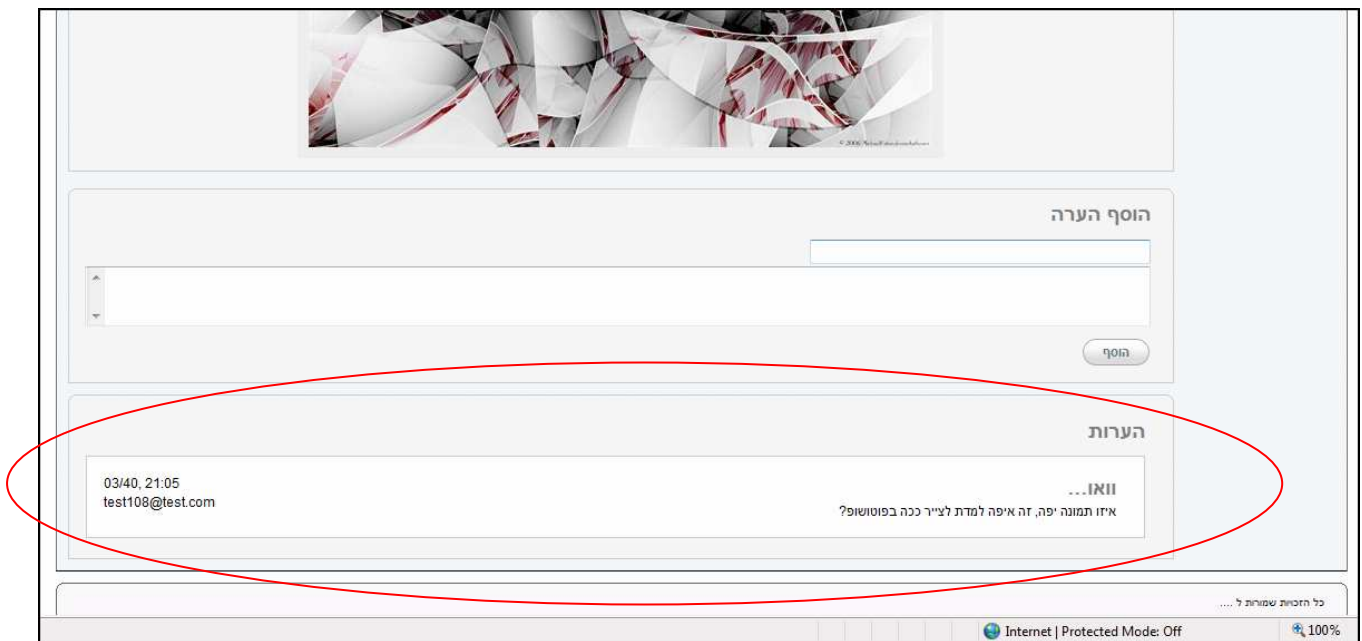
בכל מסך במערכת בו מוצגות יצירות, ניתן ללחוץ על התמונה עצמה ולהגיע לעמוד היצירה:



בגלילה למטה מוצג למשתמש ממשק הוספת הערה:



לחיצה על לחצן "הוסף" – וההערה מתווספת מיד לרשימת ההערות :



7.4. ניתוח יעילות

7.4.1. ביצועי מערכת

הבדיקות הבאות בוצעו על ידי מדידת זמן הביצוע בצד השרת בלבד, ללא זמני שליחה או קבלה של נתוני תקשורת. בדפדפן מרוחק זמני שליחה והקבלה ישתנו על בסיס קצב החיבור של המשתמש והעומס באותו זמן.

עמוד ראשי - ~800ms

עמוד "כל היצירות" - ~700ms

עמוד "היצירות שלי" - ~300ms

הוספת הערה - ~700ms

עמוד יצירה - ~200ms

התחברות - ~200ms

שמירה למועדפים והצבעה - ~100ms

7.4.2. עלות מערכת

למערכת בסביבת ייצור נדרש שרת אחד לפחות, עם 4GB זיכרון RAM. בסיס הנתונים עומד כרגע על 5MB, ויזדקק לכ-1MB מוערכים לפעילות ממוצעת של כל 100 משתמשים נוספים.

מערכת האחסון של ה-MediaServer (אחסון התמונות עצמן) מצריכה כ-1MB לתמונה ממוצעת.

על פי נתונים אלו, בחישוב קצב גדילה של כ-500 משתמשים בחודש בשנה הראשונה, וכ-10 תמונות לכל משתמש – מדובר על כ-6000 משתמשים, 60000 תמונות, משמע - כ-60GB+ של אחסון.

גיבויים יומיים ושבועיים של מסד הנתונים יצריכו עוד כ-100MB.

מסקנה - מומלץ שטח אחסון של כ-200GB תוך שקלול סיכון של פי-3 מפעילות מוערכת.

נדרש חיבור אינטרנט מהיר של לפחות 10MB upstream, 10MB downstream.

מערכות תוכנה שיש לרכוש -

Windows Server 2003

SQL Server Enterprise Edition

7.4.3. אמינות (איכות) המערכת

המערכת הינה אמינה וללא סטיות.

7.4.4. שלמות המערכת

המערכת מבצעת את כל המטרות שהוגדרו לה.

7.4.5. ייחודיות ומקוריות

המערכת מציגה ממשק וחוויית משתמש ייחודיים, בנוסף לתשתית איכותית לגדילה בהמשך.

ממשק המשתמש מעוצב באופן ייחודי ומושך, תוך מתן דגש על עיצוב נקי והבלטת יצירות המשתמש בעמוד.

המערכת מבוססת על הטכנולוגיות החדשות ביותר, לדוגמת Linq ו-WCF כמו כן, המערכת מכילה רכיבי תצוגה רבים אשר פותחו כ- Server Custom Controls וניתנים לשימוש חוזר באופן קל ומהיר.

7.5. אבטחת מידע

- מנגנון ההרשאות והמשתמשים במערכת מבוסס על Winforms Authentication ו- Asp.Net Membership.
- המערכת חוסמת גישה לעמודי משתמש פרטיים באמצעות בדיקת הרשאות בעת כניסה לעמוד.
- שימוש ב- AntiXssLibrary – קידוד פלט שנובע מה-DB באמצעות HTML ENCODING - מניעת Cross Site Scripting.
- SQL Injection בלתי אפשרי במערכת – כל השאילתות מתבצעות באמצעות פרמטרים דרך תשתית Linq. קלט לא חוקי יגרום לשגיאת מערכת עבור בקשת המשתמש הנוכחית בלבד (נסיון ה"פריצה").
- Asp.Net מסן באופן חלקי קלט משתמש בשכבת ה-UI לפני שזה מגיע לשכבות פנימיות יותר במערכת.
- מסד הנתונים מוגן באמצעות שם משתמש וסיסמה.
- חשבון המשתמש באמצעותו מתחברת המערכת למסד הנתונים הינו חשבון משתמש מוגבל, בעל הרשאות קריאה כתיבה לטבלאות בלבד. משתמש זה אינו מורשה לבצע פעולות ניהול (כגון מחיקת טבלאות, יצירה/מחיקה של משתמשים וכדומה).
- חשבונות המשתמש תחתיו רץ תהליך ASP.Net ו-IIS הינם חשבונות מוגבלים – הרשאות קריאה בלבד למערכת האחסון. (מלבד באיזור מוגדר מראש לשמירת תמונות – רכיב ה-MediaServer)

7. התוצר

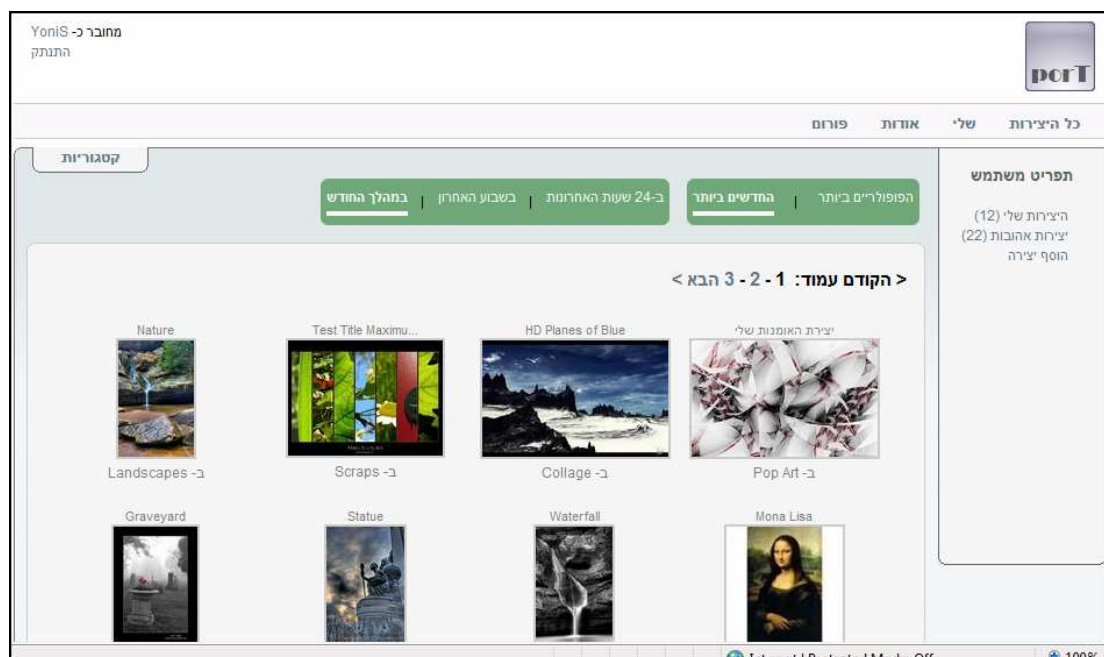
המערכת מתחלקת ל-2 רכיבי תוכנה עיקריים -

- אפליקציית המערכת הראשית – זו משרתת ומציגה את העמודים הדינמיים בהם צופה המשתמש ומכילה את לוגיקת המערכת.
- שרת ה-Media - אפליקציה נפרדת אשר משרתת את האפליקציה הראשית, ומאפשרת שמירה, שליפה ומחיקה של תמונות מאתר אחסון כלשהו.

התשתיות הנדרשות להפעלת המוצר הן :

- חיבור גישה לאינטרנט.
- שרת Windows Server 2003, עם IIS, .Net Framework 3.5.
- מסד נתונים port על גבי Sql Server 2005. שם משתמש וסיסמה מוגדרים לשימוש האפליקציות, בעלי הרשאות קריאה וכתיבה לטבלאות.
- Virtual Directory מוגדר ב- IIS המופנה לספריית האפליקציה
- הרשאות קריאה מספריית האפליקציה.
- ספריית Media מוקצת ונגישה לכתיבה לרכיב ה- MediaServer, אשר חשופה גם היא באמצעות Virtual Directory ב- IIS בו מותקן רכיב ה- MediaServer.

אתר המערכת :



רכיב ה- WebService של ה- MediaServer :

MSWS

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [Delete](#)
- [GetImageDuplicates](#)
- [GetMediaBytes](#)
- [Store](#)
- [TestDB](#)
- [TestIO](#)

This web service is using <http://tempuri.org/> as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. <http://tempuri.org/> is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

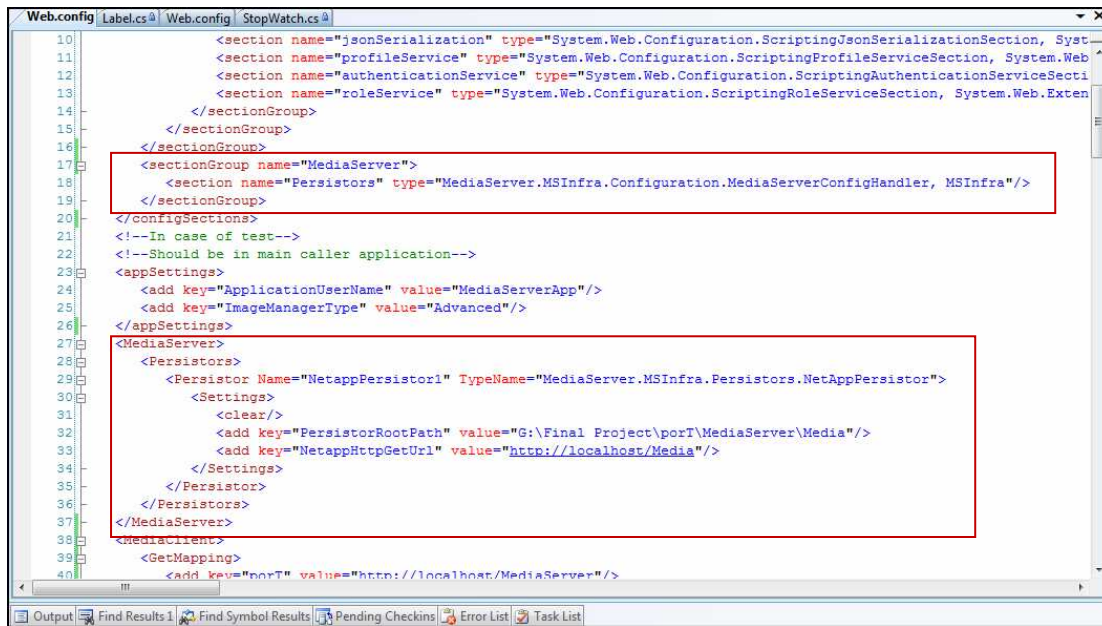
For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

```
C#  
[WebService(Namespace="http://microsoft.com/webservices/")]
```

הגדרות – Media Client :

```
Label.cs Web.config StopWatch.cs  
6 <section name="scriptResourceHandler" type="System.Web.Configuration.ScriptingScriptResourceHandlerSection,  
7 <sectionGroup name="webServices" type="System.Web.Configuration.ScriptingWebServicesSectionGroup, System.Web  
8 <section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializationSection, Syst  
9 <section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web  
10 <section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenticationServiceSecti  
11 <section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection, System.Web.Extens  
12 </sectionGroup>  
13 </sectionGroup>  
14 </sectionGroup>  
15 <section name="MediaClient" type="MediaServer.MSProxy.Configuration.MediaClientConfigHandler, MSProxy"/>  
16 <section name="loggingConfiguration" type="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSet  
17 </configSections>  
18 <MediaClient>  
19 <GetMapping>  
20 <add key="port" value="http://voni83.ath.cx/MediaServer"/>  
21 </GetMapping>  
22 <Settings>  
23 <add key="MediaServerEnviromentName" value="port"/>  
24 <add key="MediaServerDefaultHandlerPath" value="http://voni83.ath.cx/MediaServer"/>  
25 <add key="MediaServerWSPath" value="http://voni83.ath.cx/MediaServer"/>  
26 <add key="UseNewLinkFormat" value="false"/>  
27 </Settings>  
28 </MediaClient>  
29 <appSettings>  
30 <add key="HomeURL" value="http://stam/port"/>  
31 <add key="maxCategoryTreeDepth" value="2"/>  
32 </appSettings>  
33 <connectionStrings>  
34 <clear/>  
35 <add name="LocalSqlServer" connectionString="data source=localhost;initial catalog=port;user id=port;password=por  
36 </connectionStrings>
```

Media Server - הגדרות



```
10 <section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializationSection, System.Web" />
11 <section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web" />
12 <section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenticationServiceSection, System.Web" />
13 <section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection, System.Web.Extensions" />
14 </sectionGroup>
15 </sectionGroup>
16 <sectionGroup name="MediaServer">
17 <section name="Persistors" type="MediaServer.MSInfra.Configuration.MediaServerConfigHandler, MSInfra" />
18 </sectionGroup>
19 </configSections>
20 <!--In case of test-->
21 <!--Should be in main caller application-->
22 <appSettings>
23 <add key="ApplicationUserName" value="MediaServerApp" />
24 <add key="ImageManagerType" value="Advanced" />
25 </appSettings>
26 <MediaServer>
27 <Persistors>
28 <Persistor Name="NetappPersistor1" TypeName="MediaServer.MSInfra.Persistors.NetAppPersistor">
29 <Settings>
30 <clear/>
31 <add key="PersistorRootPath" value="G:\Final Project\porT\MediaServer\Media" />
32 <add key="NetappHttpGetUrl" value="http://localhost/Media" />
33 </Settings>
34 </Persistor>
35 </Persistors>
36 </MediaServer>
37 <MediaClient>
38 <GetMapping>
39 <add key="port" value="http://localhost/MediaServer" />
40 </GetMapping>
41 </MediaClient>
42 </config>
```

8. סיום

8.1. סיכום ומסקנות

תהליך העבודה בפרוייקט דמה לתהליך ה-Waterfall, כאשר רוב המערכת הייתה מוגדרת מראש לפני שלב המימוש. במהלך מימוש הפרוייקט למדתי על טכנולוגיית Linq, העמקתי את הידע ב-CSS, וכמו כן מימשתי מספר מחלקות בהן אוכל לעשות שימוש עתידי.

אחת המסקנות מהפרוייקט הינה שתהליך העבודה אשר נכפה ברובו על הסטודנט כתהליך WaterFall (כתיבת תוכן מלא למערכת לפני מימוש) הינו מיושן, וכיום קיימות מתודולוגיות פיתוח חדשניות יותר כגון Agile.

מסקנה נוספת הינה שרוב התייעוד לא שימש אותי להשגת מטרות הפרוייקט, וברובו היה מיותר במערכת מסוג זה. כמו כן, ישנו מספר רב של סעיפים במסמכים השונים אשר אינו רלוונטי לכל סוג מערכת, ולטעמי יש לאפשר חופש רב יותר בניסוח המסמכים המלווים את הפרוייקט.

המערכת כיום מומשה ונותרו מספר באגים בודדים אשר יפתרו עד הגשת המערכת.

8.2. פיתוחים עתידיים והמשך עבודה

המערכת כפי שהיא כיום היא בעלת מעט מידי תוכן על מנת להוות מערכת אמיתית. אני מתכנן להמשיך ולהרחיב את היכולות ואת חווית המשתמש, ולחשוף את המערכת לקהל הרחב כאשר תהיה בשלה לכך.

הנושאים שלדעתי יש להרחיב – פרופיל המשתמש ודירוג משתמשים, הרחבת מערכת הניקוד, שיתוף יצירות ושליחה לחברים, התראות למשתמש ב-email, העלאה של תמונות למכירה ומכירת הדפסים, ועוד.

8.3. רשימת מקורות

המקורות הרלוונטיים לכל נספח מצויינים בסופו.

9. נספחים

9.1. מסמך דרישות - SRD – Software Requirements Document

"porT" – community art portal – Software Requirements Document

1 Introduction

1.1 Purpose

The purpose of this document is to specify and document the functional requirements for the software system, namely "porT", planned for development as part of the Academic College for Engineering B.Sc. program final project.

This document will provide a basis for:

- a. Planning and system design.
- b. Test planning.

1.2 Scope of the software

The system will provide a sound basis for a community site, in which users from the field of visual design may participate, interact and contribute tutorials, creations, comments and professional tips, as a means to leverage each others' work, or just "show off" their capabilities.

1.3 Definitions, acronyms and abbreviations

Item/submission – a creation (image) a user has uploaded.

1.4 References

Project STP and SPMP documents – these documents include the test plan and project management documentation relevant to the current project.

1.5 Overview of the document

This document is mainly aimed for the developers and technical staff involved in the project, and will provide a basic understanding of the requirements to be met on project completion. The main points of interest for a developer would be:

1. The functional requirement listing, in section 3, interface requirements and the security and performance requirements near the end of that section.
2. The model description in section 2.7.

For the IT personnel, the primary points of interest are:

1. Environment considerations – section 2.4
2. Resource requirements – sections 3.4, and
3. Security requirements – section 3.6, with emphasis on backend security (related to machine and server deployment and configuration).

2 General Description

2.1 Relation to current projects

There is no relation to any other projects the developer is currently involved with.

2.2 Relation to predecessor and successor projects

There is no relation to any predecessor or successor projects.

2.3 Function and purpose

Main features are:

1. Registration of a new account – username and password
2. Personal account management
3. Storage of images
4. Portfolio view – management of stored images
5. Public portfolio view – viewing the complete portfolio of another user
6. Public view – browsing public images
7. Scoring Module – image and creator scoring
8. Commenting
9. Forum
10. "Joint Venture" – two or more people working on the same item.
11. Favorites module – saving favorite creations/ artists.
12. "Newly added" view – display artwork waiting for community rating
13. "Portfolio Printout" – automatically create a designed printed portfolio

2.4 Environmental considerations

The product will be accessible via browser (common browsers – preliminary support will include IE6/7), from anywhere in the world.

The target audience for this product is the professional/ hobbyist designer or graphical design artist, and students or applicants of the visual design academic major.

Users will be divided into two roles – Visitors and Users.

Visitors will not be able to perform actions that require a relation to an account – i.e. - uploading images, writing comments, etc.

Administrative tasks will currently be performed manually against the database by the development team, until the need for non-technical staff to perform data alterations will emerge.

The product will be run on a single server machine, running both the ASP.Net application process and the SQL server process. Hardware architecture expansion requirements will be

accommodated depending on usage and load parameters, and will be decided upon at due time.

Specifications for the target server machine (if project will be deployed for production purposes):

- x86 processor based .
- At least 1GB of memory
- IIS installed
- Operating system - Windows Server ®. (revision to be decided upon)
- Maintained by developer – accessible remotely via remote desktop

2.5 Relation to other systems.

No external systems or subsystems.

2.6 General constraints.

The project provided is intended as a proof-of-concept only, and will not be publicly available on the internet.

As such, the hardware committed to the product will not allow product support for many simultaneous users (more than 10). The product will be graphically designed to some extent but the development effort will be put into functionality and features as a priority.

Due to financial constraints:

- The database will be a freely distributable copy of the SQL Server 2005 server (namely – SQL express).
- The server on which the application will run will be the developer's home PC or his job-assigned laptop.

Time constraints are depicted in the SPMP document relevant to this project.

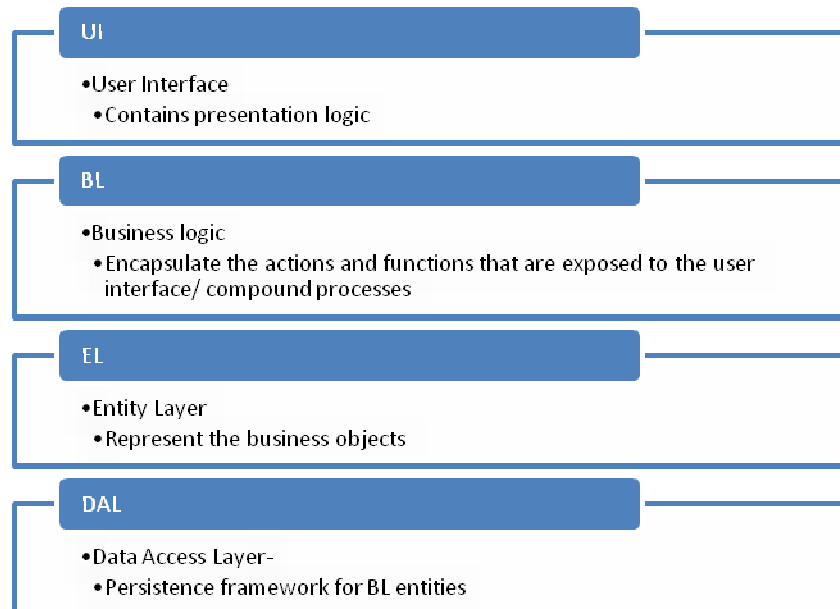
Project completion and submission date (external constraint)
- 15.5.2008.

No technological constraints are currently relevant to the development of this project.

2.7 Model description.

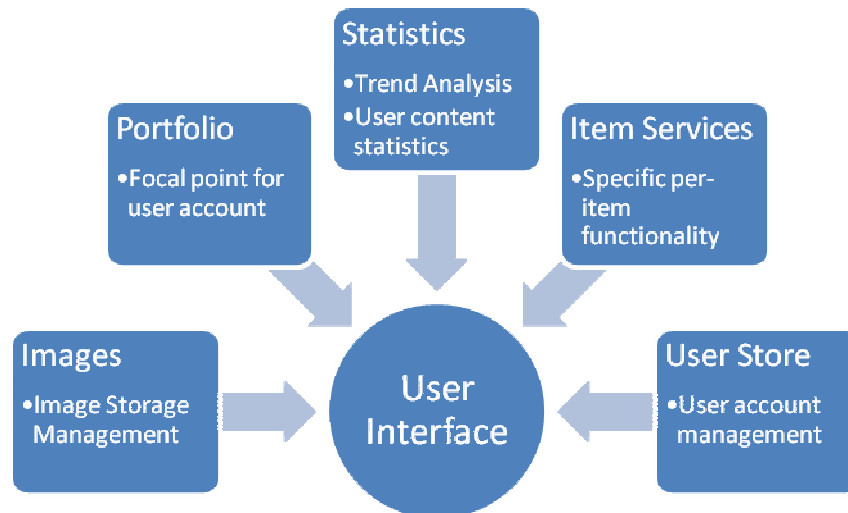
The software architecture used for the product mainly follows common object-oriented implementation idioms and is divided into logical software layers, each one with its own scope of responsibility, as shown in Figure 1.

Figure 1 – The different application layers



The business logic layer will be divided into different modules, each one encapsulating a different contextual scope, as shown in Figure 2.

Figure 2 – primary BL modules



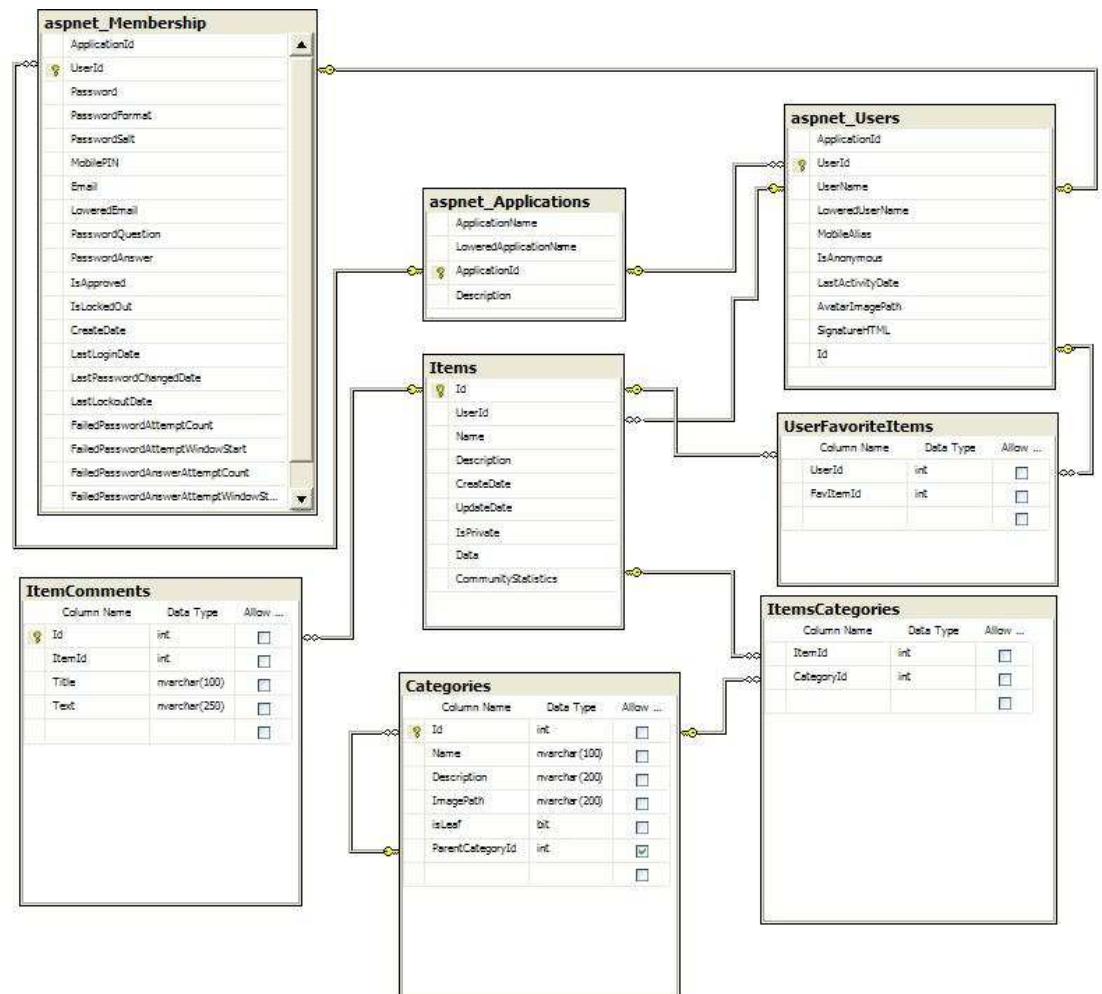
Each BL module will contain the relevant business logic functionality and will encapsulate persistence and relationship binding of business entities.

Meaning – No persistence methods (Save/Delete/Update/Get) or relationships (one-many, many-on, one-one) will be created on the frontend layer. These services will be provided by the BL and hidden from presentation implementation considerations.

Architecture Comments :

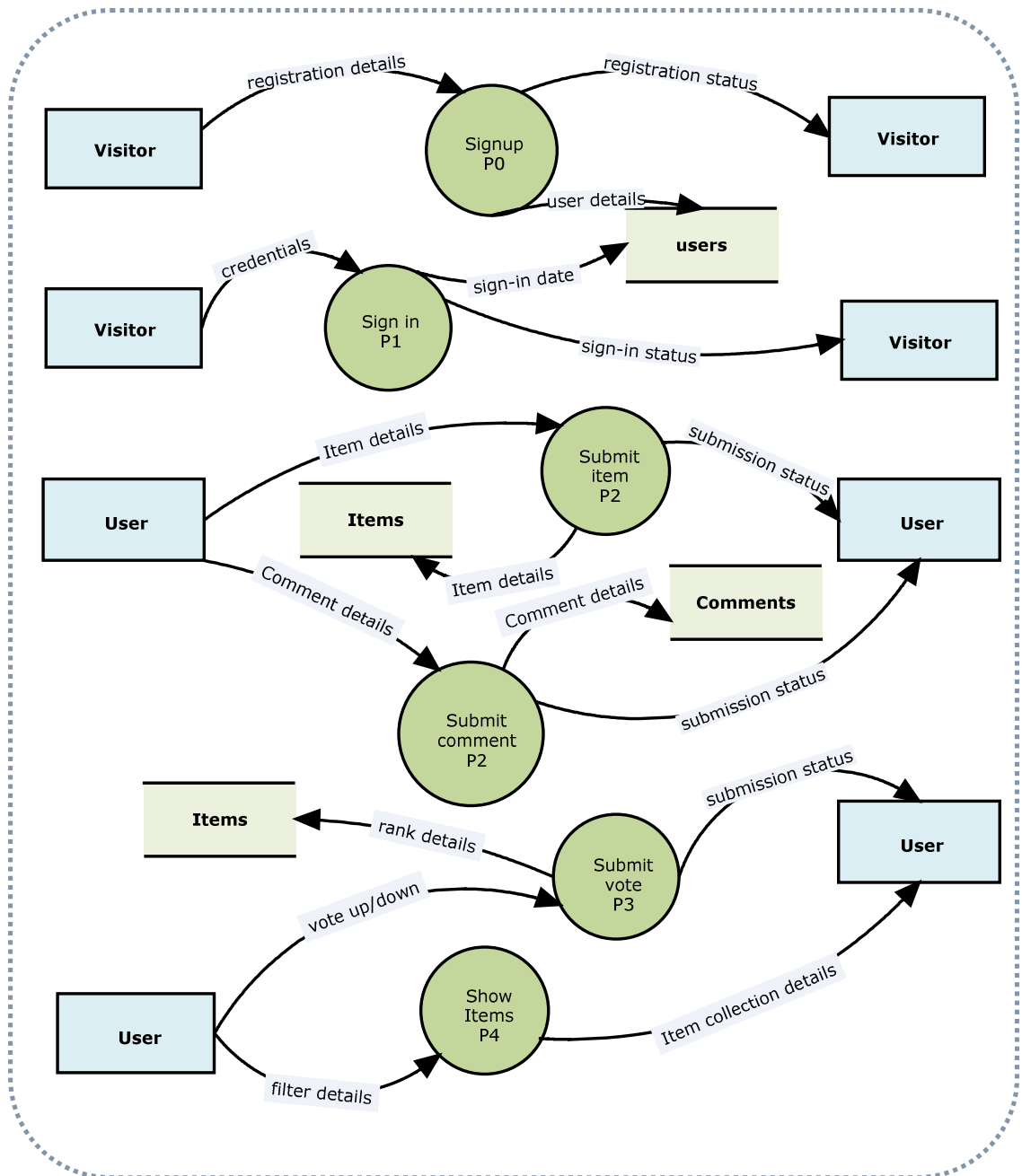
- All persistent and non-persistent entities should be exposed to the frontend layer by proper Interfaces (User exposed as IUser).
- Entity instances will be created using a smart factory, to minimize code refactoring when changing construction pattern.
- All BL, EL and DAL functions are context blind – no references in code to frontend contextual elements (User.Current, etc.) – these are received as parameters from the external layer. (testing and coupling considerations)

Following is an ERD schema, presenting the main entities in the application:



Following is a DFD diagram, displaying some of the functionality available in the application (will be expanded during a feature based iterative development process):

DFD 0 - porT



3 Specific Requirements

3.1 Functional requirements.

1. Account registration – The web application will display personal and public information with regard to whether the viewing user is signed in as a member. In order to be a member of the site the user will be able to sign up using a personal email, a password and profile information (Country, Age, others to be decided later on).

After registration is complete – the user will be signed in to the system, until session expiration occurs. The user will be in a non-activated state, which will prevent certain functionality such as messaging, and others (to be decided later).

After registration is complete, an email will be sent to the user's email address confirming his registration, and requesting activation of the account by visiting an activation page on the site with a special activation key, which will reside in the link (automatically created upon email generation).

The user will then be redirected to the homepage and a message will inform him that his account is now active.

2. Personal account management – if a user is signed **in to** his account, he will be able to access a special section devoted to personal account management.

In this section he will be able to change his email address, his password, and personal profile information.

In addition to these, the personal account management section will include a sub section which will allow the user to control system usage preferences – how and when will the system contact him, by email or otherwise, if other users in the system can see his personal profile information, etc.

3. A registered user will be able to add to and delete images from his portfolio. All new submissions will be automatically set to be publically viewable, until the user states otherwise.

All new submissions will be added to a special "new" status, in which they will be displayed in a special section of the website dedicated to newly added creations, waiting for community rating.

A creation will change status to "porTed" when it will receive a rating threshold calculated using -

- The number of votes by the community
- The amount of time this creation is considered "new"
- The number of times it was viewed by the community

4. Portfolio view – The main personal portfolio page, in which the user will be able to add, delete and update existing submissions,

and view relevant information to his portfolio.

The portfolio view will include a list of all user submissions, hints to new community content and rating information relevant to his submissions, and will be able to navigate to this content.

5. Similar to portfolio view but in the context of a public portfolio of another user – editing options will be hidden and community content will be editable. (Add comments, add to favorites, etc.)
6. The public view (Homepage) will default to all public items, sorted by popularity (rank).
The view will enable sorting by recency, popularity, and filtering by category.
7. Each vote on an item will be added to the rank of that submission and to the total score of the owner, and will be used as a means of sorting when displaying the most popular items. The rank of the voter himself affects the impact his votes have on other users (professional opinions matter more).
8. Users will be able to comment on each others' work, and comments can be rated as useful or not useful.
There will be a "recent opinions" page displaying newly added comments in the context of the item, in which users will be able to navigate to the item itself.
9. The forum will be an external 3rd party component (forum.porT.co.il), in which users can create topics of conversation, and link these topics to items in the application. Contests, community events, release notes etc. will be publicized on the forum.
10. A user can ask for "help" on a creation, if he uploads the source .psd file (Photoshop file), and other users can "pick up" the work and add revisions to it. Each revision can have its own sub-revisions, and in this way a "work in progress" can be performed by multiple artists.
11. A signed-in user will be able to save other users or public items to his "favorites". Items that have been moved to private by their owners will be marked as "removed from public view" in the "Favorites" page.
Each addition to the user's favorites will be counted on the item or user that has been saved, and the rank for that item will be raised.

12. The "newly added" page will be a page containing all recently added items that have not met the minimum "exposure" limit (votes/ favorites/ amount of time since submission). These items are a concentration of all items pending community contribution of comments, votes etc.
13. A user will be able to print his portfolio into a well-designed and layed-out printout, which he will be able to use when submitting a job or study application.
14. Currently, administrative tasks will be performed by the developer manually on the database. Support for an administrative interface will be considered later on, during the maintenance phase of the product.

3.2 Performance requirements.

Frontend performance:

- Session creation (first request) will not take longer than 1 second.
- Page load time across the application will not rise above 2.5 seconds (after first request)

Backend performance:

- Database table fields will be properly indexed to optimize seek time on queries.
- Textual queries will be performed against full-text catalogues. (SQL Server 2005 feature)

3.3 Interface requirements

Currently, there are no interfaces to other systems to be defined.

User interface requirements:

1. User interface design will be consistent across the application, and styles will be set by external CSS (Cascading Style Sheet) files.
2. Design philosophy will be minimalistic, following the idea that each page should comprise of a maximum of 3 data contexts –

main, contextual, and supplementary.

The main data context is the information targeted for this view, i.e. – for portfolio view, the main data context contains the portfolio items.

Contextual data in this example may include a title with my username, how many items I have, what is my current rating, etc.

Supplementary data – other users who have saved my portfolio to their favorites, etc. The design should be well thought of in order to minimize supplementary data.

3. Item information will be categorized – sorting and filtering options should be supplied only for main data context objects.
4. Lists of many items should include a pager, and the amount of pages will not surpass 20.
5. Input validation should be supported across the application and proper error messages should be displayed. It is preferred to perform validation on client-side.
6. All actions within the scope of the currently viewed context should be performed asynchronously (without complete page refresh actions), within reasonable and feasible limitations.

3.4 Resource requirements.

Development environment resource requirements:

- IDE – Visual studio 2005
- Web server – IIS 6
- Development system:
 - Windows XP
 - 2GB ram
 - High-end processor (Intel dual core 2.2Ghz and upwards)
- Network and internet connectivity

Production system resource requirements
(will not be included as part of proof-of-concept):

- Web server – IIS 6
- .Net Framework 2.0
- ASP.Net Ajax redistributable deployed
- SQL Server 2005
- **Production system**
 - 4GB Ram
 - 2x Xeon 2.8Ghz processors
 - Windows Server 2003
- Network and internet connectivity, a static IP address, and a global DNS registration of the domain www.porT.co.il and www.porT.com to said static address.

3.5 Verification requirements.

1. All methods in the BL and EL will be tested by creating a Unit test for each.
2. Backend verification will be evaluated by the successful execution of all unit tests, and 100% code coverage of backend sub-projects.
3. Frontend verification will be carried out by the developer according to a test plan described in the project's STD.

3.6 Security requirements.

A lack of sufficient security measurements may provide for a system which is susceptible to external attacks, such as data deformation and corruption, identity theft, phishing scams, application unavailability and privacy infringements.

Since the application is a web application, it is externally available to all users on the web. The following are measures which should be employed to minimize these risks -

- Frontend :
 - Users will not be able to access private information on other members of the application, and will not be able to edit content created by others, unless explicitly stated otherwise as part of a functional requirement of an application feature.
 - Passwords will obey a strength policy, which will be defined later under usage and user experience limitations.
 - User input should be validated treated in it's HTML encoded form. (i.e. – the "<" character should be saved in the database, if needed, as "<").
- Backend :
 - Data stored in any storage medium, will be protected with secured credentials.
 - The server will run the ASP.Net and IIS process under constrained accounts, with proper permission settings on both, to avoid privilege escalation in case of a frontend breach.

SDD – Software Design Description - מסמך תיכון .9.2

"porT" – community art portal – Software Design Description

1. Introduction

1.1. Purpose

The purpose of this document is to specify all aspects of the software system design for the "porT" web application project.

The main audience for this document is software developers, basing system implementation on the application interfaces and functional flows laid out in this document.

This document will also provide sections aiding management or external consulting staff by describing system architecture, referenced components and modules and system interfaces, by which a general understanding of the system may be attained.

1.2. Scope of the software

The system will provide a sound basis for a community site, in which users from the field of visual design may participate, interact and contribute tutorials, creations, comments and professional tips, as a means to leverage each others' work, or just "show off" their capabilities.

1.3. Definitions, acronyms and abbreviations

- Item/submission – a creation (image) a user has uploaded.
- (X:Y) – Attachment reference abbreviation
(x: median report number 1/2, y: section number)
- Poi – point of interest

1.4. References

Project SRD (2:a), STD (2:c) and SPMP (2:d) documents – these documents include the requirements, test plan and project management documentation relevant to the current project.

1.5. Overview

This document is mainly aimed for the developers and technical staff involved in the project, and will provide the detailed specification of the system design, including application layers and modules and their application interfaces (API).

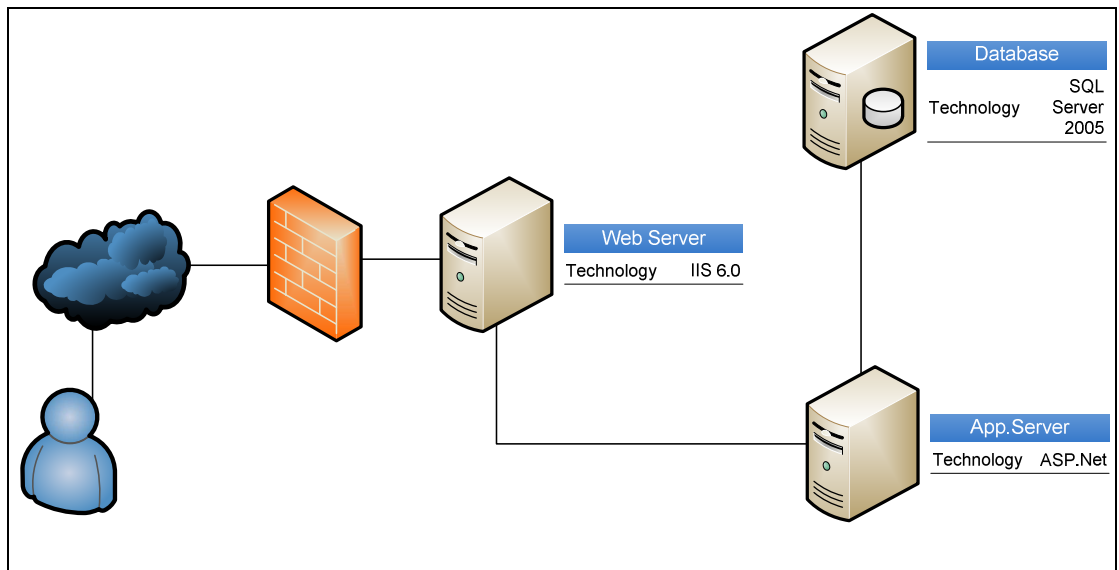
The main points of interest for management and IT personnel would be sections 2.1 – System architecture, and section 4 – user interface design.

The main points of interest for developers would be section 3 – detailed component description (said "API").

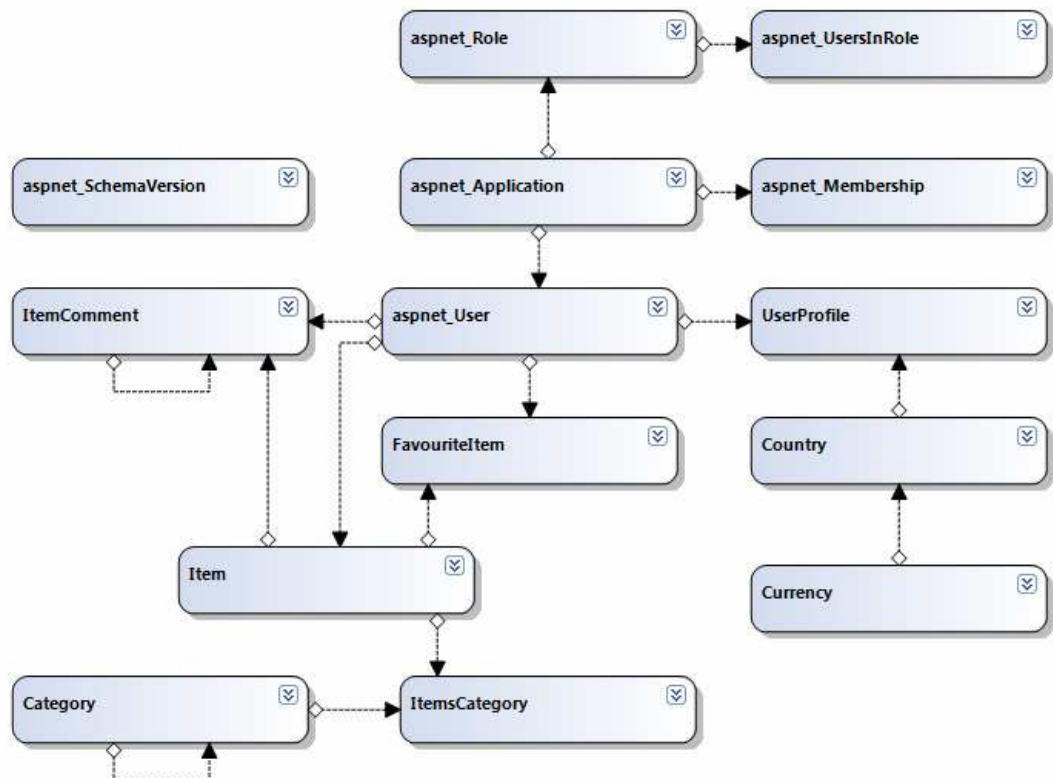
2. System Architectural Design

2.1 Chosen System Architecture

Figure 2.1 Machine-per-Server layout (theoretical view)

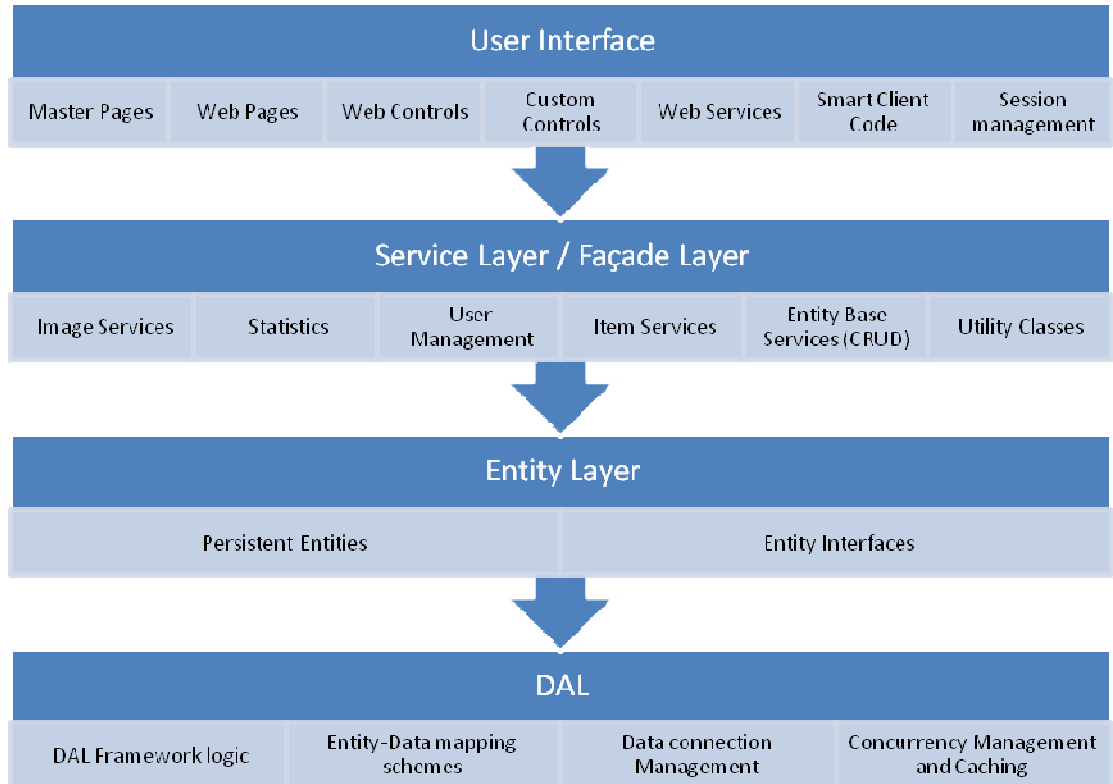


Primary entity schema :



The software architecture used for the product mainly follows common object-oriented implementation idioms and is divided into logical software layers, each one with its own scope of responsibility, as shown in Figure 2.1

Figure 2.1 – Software System Layers



The system is composed of four main layers, each one with its own responsibility.

Following are the main poi's:

a. User Interface

The user interface layer is responsible for delivering web pages and controlling their server side logic. It includes the following components :

1. Master Pages – includes header/footer and shared script and styling references. These are included in each and every page.
2. Web Pages – specific content pages, mainly used to control content layout and contain cross-control communication, and page-level logic.
3. User Controls/ Custom Controls – domain specific components which supply variable functionality, i.e. Image gallery, file uploader, etc.
4. Web Services – exposing a common HTTP API for 3rd party software and JavaScript interaction with the application server. Mainly used in the system for AJAX capabilities.

5. Smart Client Code – JavaScript files for client side functionality, includes browser DOM manipulation methods, DHTML animations, client side validation scripts and more.
6. Session Management – client-server sessions are managed in this layer and provide functional continuity and authentication persistence.

b. Service Layer

1. Base Services – a base class for all service classes, enabling create/read/update/delete (CRUD) operations on all entities. Class is generic and crud methods are overridable.
2. Image Services – a service component responsible for managing the persistence of images, getting image links for display on browser and resizing images before display.
3. Statistics – exposes statistical information relevant to items and users, such as – number of comments, popularity rating, etc.
4. User Management – responsible for creating and deleting users and managing user preferences and information.
5. Item Services – responsible for creating, updating and deleting items, adding comments to items, saving favorite items to a users account etc.

c. DAL

1. Wraps ORM framework logic.
2. Manages database connections and persistent object concurrency.
3. Contains mapping schemes for ORM framework.
4. Provides 1st level cache (by ID, during single request) and transactionality.

2.2 System Interface description

The system has a single external interface which is the user interface (web interface).

3. Detailed Description Of Components

3.1. DAL :

```
public static class DataContextWrapper<T> where T : DataContext
```

3.1.1 Type – Class

3.1.2 Purpose

Provide a singleton wrapper to the `DataContext` object. Forces singularity of `DataContext` instance per http request (instance = database connection).

3.1.3 Function inputs and outputs

```
public static T Instance –
```

retrieves an instance of T (a `DataContext` object) if hasn't already been created for the current http request. If an instance has already been created – the property returns that instance.

3.1.4 Interfaces

All service-layer components access the static Instance property in order to get the active request `DataContext` object, and use it to query, insert, update or delete records from database tables.

3.1.5 Data

This a wrapper to a .Net Linq `DataContext` object. This object provides a queryable interface to database table data.

3.2. DAL :

```
public partial class portDataContext : System.Data.Linq.DataContext
```

3.2.1 Type – Class

3.2.2 Purpose

Provides central data access to all database tables, and exposes a queryable interface specific to the database schema exposed by the PORT database.

3.2.3 Function inputs and outputs

Table interfaces:

```
public System.Data.Linq.Table<aspnet_Application> aspnet_Applications
```

```
public System.Data.Linq.Table<UserProfile> UserProfiles
```

```
public System.Data.Linq.Table<aspnet_Membership> aspnet_Memberships
```

```
public System.Data.Linq.Table<aspnet_Role> aspnet_Roles
```

```
public System.Data.Linq.Table<aspnet_User> aspnet_Users
```

```
public System.Data.Linq.Table<aspnet_UsersInRole> aspnet_UsersInRoles
```

```
public System.Data.Linq.Table<Category> Categories
```

```
public System.Data.Linq.Table<Country> Countries
```

```

public System.Data.Linq.Table<Currency> Currencies
public System.Data.Linq.Table<FavouriteItem> FavouriteItems
public System.Data.Linq.Table<ItemComment> ItemComments
public System.Data.Linq.Table<Item> Items
public System.Data.Linq.Table<ItemsCategory> ItemsCategories

```

3.2.4 Interfaces

A service layer methods may take a instance of this object, and user the table interfaces to query the database for entities. Linq-To-SQL technology allows the following syntax to be used in C# code (for instance – a query for items belonging to a user, using a given email as a parameter):

```

porTDataContext dc = DataContextWrapper<porTDataContext>.Instance;

var query = from i in dc.Items
             join u in dc.aspnet_Users on i.UserId equals u.Id
             join m in dc.aspnet_Memberships on u.UserId equals m.UserId
             where m.Email == email
             select i;

return query.ToList<Item>();

```

3.2.5 Data

Uses the System.Data.Linq.Table class to provide the queryable interface.

3.3. EL :

```
public partial class aspnet_Application
```

3.3.1 Type – Entity Class

3.3.2 Purpose

Entity – differentiates Membership applications. The database Membership schema supports several client applications, each one references by a different aspnet_Application record.

3.3.3 Function inputs and outputs

N/A

3.3.4 Interfaces

Used while querying for users and membership information relevant to the current application.

Exposes the following hookable events:

```

public event PropertyChangedEventHandler PropertyChanged;
public event PropertyChangedEventHandler PropertyChanged;

```

3.3.5 Data

```
public string ApplicationName
public string LoweredApplicationName
public System.Guid ApplicationId
public string Description
public EntitySet<aspnet_Membership> aspnet_Memberships
public EntitySet<aspnet_Role> aspnet_Roles
public EntitySet<aspnet_User> aspnet_Users
```

3.4. EL :

```
public partial class UserProfile
```

3.4.1 Type – Entity Class

3.4.2 Purpose

This entity contains additional information on each user.

3.4.3 Function inputs and outputs

N/A

3.4.4 Interfaces

Used in personal account management features, general application-to-user behavior and perhaps in future localization scenarios.

3.4.5 Data

```
public System.Nullable<int> Age
public bool AllowEmails
public aspnet_User aspnet_User
public Country Country
```

3.5. EL :

```
public partial class aspnet_Membership
```

3.5.1 Type – Entity Class

3.5.2 Purpose

This entity is used by the membership mechanism, primarily for authentication, password management and email purposes

3.5.3 Function inputs and outputs

N/A

3.5.4 Interfaces

Is used during user authentication, and other user administration tasks.

3.5.5 Data

```
public string Password
public int PasswordFormat
public string PasswordSalt
public string Email
public string LoweredEmail
```

```

public string PasswordQuestion
public string PasswordAnswer
public bool IsApproved
public bool IsLockedOut
public System.DateTime CreateDate
public System.DateTime LastLoginDate
public System.DateTime LastPasswordChangedDate
public System.DateTime LastLockoutDate
public int FailedPasswordAttemptCount
public System.DateTime FailedPasswordAttemptWindowStart
public int FailedPasswordAnswerAttemptCount
public System.DateTime FailedPasswordAnswerAttemptWindowStart
public string Comment
public aspnet_Application aspnet_Application

```

3.6. EL :

```
public partial class aspnet_Role
```

3.6.1 Type – Entity Class

3.6.2 Purpose

This entity defines user roles, which may later relate to user privileges.

3.6.3 Function inputs and outputs

N/A

3.6.4 Interfaces

Used while validating a request for security privileges on resource.

3.6.5 Data

```

public string RoleName
public string LoweredRoleName
public string Description
public EntitySet<aspnet_UsersInRole> aspnet_UsersInRoles
public aspnet_Application aspnet_Application

```

3.7. EL :

```
public partial class aspnet_User
```

3.7.1 Type – Entity Class

3.7.2 Purpose

This entity represents a registered user.

3.7.3 Function inputs and outputs

N/A

3.7.4 Interfaces

Used in all user-related services, and is a references by all user-related entities (user comments, user images, user favorite items, etc..)

3.7.5 Data


```

public string UserName
public string LoweredUserName
public string MobileAlias
public bool IsAnonymous
public System.DateTime LastActivityDate
public EntitySet<UserProfile> UserProfiles
public EntitySet<FavouriteItem> FavouriteItems
public ItemComment ItemComment
public EntitySet<Item> Items
public aspnet_Application aspnet_Application

```

3.8. EL :

```
public partial class Category
```

3.8.1 Type – Entity Class

3.8.2 Purpose

Represents the main taxonomy, dividing items into groups according to theme or context.

3.8.3 Function inputs and outputs

N/A

3.8.4 Interfaces

Mainly used during navigation.

3.8.5 Data

```

public string Name
public string Description
public string ImagePath
public bool isLeaf
public int ParentCategoryId
public EntitySet<Category> Categories
public EntitySet<ItemsCategory> ItemsCategories
public Category ParentCategory

```

3.9. EL :

```
public partial class Country
```

3.9.1 Type – Entity Class

3.9.2 Purpose

This entity represents a country

3.9.3 Function inputs and outputs

N/A

3.9.4 Interfaces

N/A

3.9.5 Data

```

public string Name
public string DateFormat
public bool HasStates
public string isoCode
public Currency Currency

```

3.10. EL :

```

public partial class Currency

```

3.10.1 Type – Entity Class

3.10.2 Purpose

Represents a currency. Maintained for future use.

3.10.3 Function inputs and outputs

N/A

3.10.4 Interfaces

N/A

3.11.5 Data

```

public string Name
public string ShortName

```

3.11. EL :

```

public partial class FavouriteItem

```

3.11.1 Type – Entity Class

3.11.2 Purpose

This entity represents a "favourite item" relationship between a user and an item

3.11.3 Function inputs and outputs

N/A

3.11.4 Interfaces

N/A

3.11.5 Data

```

public aspnet_User aspnet_User
public Item Item

```

3.12. EL :

```
public partial class ItemComment
```

3.12.1 Type – Entity Class

3.12.2 Purpose

A relational entity, meant to keep the connection between an item and it's comments (n:m)

3.12.3 Function inputs and outputs

N/A

3.12.4 Interfaces

N/A

3.12.5 Data

```
public string Title
public string Text
public System.DateTime CreateDate
public System.DateTime UpdateDate
public System.Nullable<int> UserId
public EntitySet<ItemComment> Replies
public ItemComment SourceComment
public aspnet_User Commenter
```

3.13. EL :

```
public partial class Item
```

3.5.1 Type – Entity Class

3.5.2 Purpose

This entity represents a single artwork record.

3.5.3 Function inputs and outputs

ImagePath represents a token to be used by the MediaServer component.

3.5.4 Interfaces

a

3.5.5 Data

```
public string Name
public string Description
public System.DateTime CreateDate
public System.DateTime UpdateDate
public bool IsPrivate
public System.Xml.Linq.XElement Data
public System.Xml.Linq.XElement CommunityStatistics
public bool IsDeleted
public string ImagePath
public EntitySet<ItemComment> ItemComments
public EntitySet<ItemsCategory> ItemsCategories
```

```
public aspnet_User Owner
```

3.14. Service Layer :

```
public class ImageServices
```

3.14.1 Type – Service Class

3.14.2 Purpose

Provides all functionality related to the management of image Items.

3.14.3 Function inputs and outputs

- `public void SaveImageToAccount(string email, HttpPostedFile file)`
Saves an image file, posted by the user, to the users' account. The account is identified by the users' email address, which is unique.
- `public List<Item> GetUserItems(string email)`
Retrieves all images uploaded by a given user. User is identified by email. Retrieves only items that are not marked as "IsDeleted".
- `public void DeleteImage(int itemId)`
Deletes an image record. Physically – only mark the item as "Deleted"
- `public Item GetItemById(int itemId)`
Retrieved an image item by it's Id.
- `public List<Item> GetPublicItem(int maxResults)`
Retrieve a list of publicly viewable items, which are not marked as "Deleted"
- `public List<Item> GetPublicItem(int userId, int maxResults)`
Retrieve a list of publicly viewable items, posted by the specified user, and are not marked as "Deleted"
- `public List<ItemComment> GetItemComments(int itemId)`
Retrieves a list of comments, posted on the requested item (identified by the itemId parameter)
- `public void AddCommentToImage(aspnet_User user, int itemId, int? parentCommentId, string commentTitle, string commentText)`
Add a comment to the specified item (identified by the itemId parameter). Initialized a new comment with the CreateDate and UpdateDate set to the current date and time.
- `public void VoteForItem(aspnet_User user, int itemId, int rating)`
Posts a vote for the specified item, from the specified user.
- `public void SaveItemToFavorites(aspnet_User user, int itemId)`
Saves the specified item the users' favorite items collection
- `public void GeneratePrintOut(aspnet_User user)`
Generates a pdf-formatted printout of the user's portfolio, and writes it to the response stream.

- `public void CheckOut/In(aspnet_User user, int itemId)`
"Checks out" and "Checks In" an image. Once an image is checked out to a certain user, it cannot be checked out again until that user checks it back in. This will be used by the "Joint Venture" feature for continuous work on a certain artwork.

3.14.4 Interfaces

N/A

3.14.5 Data

N/A

3.15. Service Layer :

`public partial class UserServices`

3.15.1 Type – Service Class

3.15.2 Purpose

Provide all functionality relating to user management.

Some functionality is already provided inherently by the Asp.Net membership mechanism, and is wrapped by designated controls (registration, for example is managed by the CreateUserWizard control).

3.15.3 Function inputs and outputs

- `public bool AuthenticateUser(string email, string password)`
Returns a true or false value, depending on the success or failure of the authentication process.
- `public aspnet_User GetUserByEmail(string email)`
Retrieve a user by his email address
- `public void DeleteUser(string email)`
Deletes a user from the system
- `public void SendConfirmationEmail(string email)`
Sends the user a confirmation email. This method is used after sign-up, and the user cannot log in until he clicks on the link provided in the confirmation email.

The link will contain a confirmationToken, used to identify the user when he completes the registration process.

- `public void ActivateWithConfirmationToken(string confirmationToken)`
Activates the account related to the confirmationToken supplied.
- `public void ResetPassword(string email)`
Resets the user's password (user is identified by his email) and sends a new password to the user's email.
- `public void SetPassword(string email, string oldPassword, string newPassword)`
Sets the user's password.

3.15.4 Interfaces

N/A

3.15.5 Data

N/A

3.16. MediaServer :

3.16.1 Type – Component

3.16.2 Purpose

This is a multi-tiered component, which provides a central, scalable persistency solution for media data, such as images and documents. For the current requirements of the project, only images are handled by this component.

3.16.3 Function inputs and outputs

Provides an abstraction layer to remotely persisted image files. The main functionality includes saving images to remote storage, retrieving image data stream from remote storage, remote replication of images, on-the-fly resize of images, and image display.

3.16.4 Interfaces

The client applications use this component via the MediaServerProxy class, as described later on in this document, in combination with a set of configuration keys set to describe the environment in which the client application resides. The client application environment determines which remote storage should be accessed, and what is the persistency method.

3.16.5 Data

N/A, see the following class descriptions.

3.17. MediaServer :

```
public class MediaServerDataContext : System.Data.Linq.DataContext
```

3.17.1 Type – Class

3.17.2 Purpose

Exposes queryable interfaces to MediaServer tables.

3.17.3 Function inputs and outputs

N/A

3.17.4 Interfaces

Is used by other MediaServer components to query the database

3.17.5 Data

```
public System.Data.Linq.Table<MarkedForDeletion> MarkedForDeletions
public System.Data.Linq.Table<Repository> Repositories
public System.Data.Linq.Table<MediaIdGenerator> MediaIdGenerators
public System.Data.Linq.Table<MediaInfo> MediaInfos
```

3.18. MediaServer :

```
public class MarkedForDeletion
```

3.18.1 Type – Entity Class

3.18.2 Purpose

Used for performing synchronization operations between database records and actual files.

3.18.3 Function inputs and outputs

N/A

3.18.4 Interfaces

N/A

3.18.5 Data

```
public string mediaToken
```

3.19. MediaServer :

```
public class Repository
```

3.19.1 Type – Entity Class

3.19.2 Purpose

A persistent entity which represents the configured link between an environment, and media type and a persistor.

One environment (i.e. – client application) may contain several repositories, ordered by priority and divided by their responsibility over media types.

Each such repository is linked to a persistor, which is a persistency medium used to physically save files to a certain location, via a certain technology.

3.19.3 Function inputs and outputs

- `public MediaPersistorBase GetPersistor()`
Retrieves a persistor implementation according to the persistor name stated in the repository.

3.19.4 Interfaces

N/A

3.19.5 Data

```
public string Name
public string EnvName
public int Priority
public byte MediaType
public string PersistorName
public MediaIdGenerator MediaIdGenerator
```

3.20. MediaServer :

```
public partial class MediaIdGenerator
```

3.20.1 Type – Entity Class

3.20.2 Purpose

One of these is kept for each repository, and they contain a counter of the mediaId, which is used to maintain uniqueness while persisting to the file-system, or any other persistence medium.

3.20.3 Function inputs and outputs

N/A

3.20.4 Interfaces

N/A

3.20.5 Data

```
public long MediaId
```

3.21. MediaServer :

```
public class RepositoryFacade
```

3.21.1 Type – Service Class

3.21.2 Purpose

Provide all functionality which is relevant to the Repository interface.

3.21.3 Function inputs and outputs

- `public static List<Repository> GetRepositories()`
Returns all repositories from the database
- `private static List<Repository> GetRepositories(string environmentName, MediaType mediaType)`
Returns all repositories attached to the requested environment, and which serve the requested mediaType. The results are sorted by priority, in descending order
- `public static string StoreAndGetMediaToken(Repository rep, byte[] mediaBytes, long mediaId, string fileName)`
Save the media stream using the persistor related to this repository, and returns a mediaToken, used to identify this media file in future references.

- `public static void Delete(Repository rep, string mediaToken)`
Deletes a media file from the repository.
- `public static byte[] GetMediaBytes(Repository rep, string mediaToken)`
Retrieve a media byte stream from the repository, according to the specified mediaToken.
- `public static Repository GetActiveRepository(string environmentName, MediaType mediaType)`
Returns the relevant repository for this environmentName and MediaType (highest priority if there is more than one)
- `public static Repository GetRepositoryByMediaToken(string mediaToken)`
Since the mediaToken contains meta-data concerning it's repository and environment, this function will find the repository in which the specified mediaToken's media file resides.

3.21.4 Interfaces

N/A

3.21.5 Data

N/A

3.22. MediaServer :

`public abstract class MediaPersistorBase`

3.22.1 Type – Abstract Class

3.22.2 Purpose

Provides an interface definition for a persistor.

3.22.3 Function inputs and outputs

N/A

3.22.4 Interfaces

- `public abstract string StoreAndGetMediaToken(byte[] mediaBytes, long mediaId, string repositoryRoot, string fileName);`
- `public abstract void Delete(string mediaToken);`
- `public abstract byte[] GetMediaBytes(string mediaToken);`

3.22.5 Data

3.23. MediaServer :

`public class FSMediaPersistor : MediaPersistorBase`

3.17.1 Type – SubClass

3.17.2 Purpose

Persists media files to the file-system

3.17.3 Function inputs and outputs

- `public FSMediaPersistor(PersistorConfiguration config)`
Constructor – received a persistorConfiguration object instance, and sets it aside in a private member for later use.
- `public override string StoreAndGetMediaToken(byte[] mediaBytes, long mediaId, string repositoryRoot, string fileName)`
Interface implementation, see interface definition, and repository definition.
- `public override void Delete(string mediaToken)`
Interface implementation, see interface definition, and repository definition.
- `public override byte[] GetMediaBytes(string mediaToken)`
Interface implementation, see interface definition, and repository definition.

3.17.4 Interfaces

N/A

3.17.5 Data

N/A

3.24. MediaServer :

`public class NetAppPersistor : MediaPersistorBase`

3.24.1 Type – SubClass

3.24.2 Purpose

Provides similar functionality as FSMediaPersistor, only that it saves to a shared drive, and retrieves from a remote virtual directory, mapped to that drive.

3.24.3 Function inputs and outputs

Same as FSMediaPersistor

3.24.4 Interfaces

N/A

3.24.5 Data

N/A

3.25. MediaServer :

`public static class MediaServerConfiguration`

3.25.1 Type – Static Class

3.25.2 Purpose

Wraps and provides access to the configuration settings section which describes persistor configuration (root paths, ip addresses etc).

3.25.3 Function inputs and outputs

- `public static Dictionary<string, PersistorConfiguration> GetPersistors()`
Get a dictionary of configuration objects, indexed by persistor name.

3.25.4 Interfaces

N/A

3.25.5 Data

N/A

3.26. MediaServer :

```
public class PersistorConfiguration
```

3.26.1 Type – Class

3.26.2 Purpose

A data object, used to describe configuration data

3.26.3 Function inputs and outputs

N/A

3.26.4 Interfaces

N/A

3.26.5 Data

```
public NameValueCollection ConfigData
public string TypeName (a fully qualified type name- may be used to reflect the persistor)
public string Name
public PersistorConfiguration(string name, string typeName, NameValueCollection configData)
```

3.27. MediaServer :

```
public class MSWS : System.Web.Services.WebService
```

3.27.1 Type – WebService

3. 27.2 Purpose

Is remotely called in the backend deployment of the mediaServer component, and wraps all the functionality which MediaServer provides, in a single API.

3. 27.3 Function inputs and outputs

```
public string Store(byte[] mediaBytes, MediaType mediaType, string fileName, string
environmentName)
public string[] GetImageDuplicates(ImageCopyRequest[] requests, string environmentName)
public void Delete(string mediaToken)
public byte[] GetMediaBytes(string mediaToken)
public string TestIO()
public List<string> TestDB()
```

3. 27.4 Interfaces

HTTP.

3. 27.5 Data

N/A.

3.28. MediaServer :

```
public class MediaServerProxy
```

3. 28.1 Type – Proxy Class

3. 28.2 Purpose

Provide a wrapper to the remote web service functionality, including some configuration resolution operations.

Functions are similar to those in interface to MSWS, but are slimmed down for the client application's comfort – the proxy Resolves the remote WebService IP and the client environmentName from configuration, and perform the WebService call to MSWS with these values.

3. 28.3 Function inputs and outputs

- `public static string` StoreAndGetMediaToken(`Image` image, `string` fileName)
- `public static void` DeleteMedia(`string` mediaToken)
- `public static string` GetImageLink(`string` mediaToken, `ImageMode` mode, `int?` size, `ImageSizeConstraint` constraint)
Returns a link that can be rendered to the client, which will in turn – create an http request from the client browser to the frontend mediaServer deployment, in order to display the image.
- `public static byte[]` GetMediaBytes(`string` mediaToken)
- `public static Image` GetImage(`string` mediaToken)
Convenience method, converting the returned stream to an image object.
- `public static string[]` GetDuplicates(`ImageCopyRequest[]` copyRequests)

3. 28.4 Interfaces

N/A

3. 28.5 Data

`ImageMode` – Full, Custom or Thumbnail

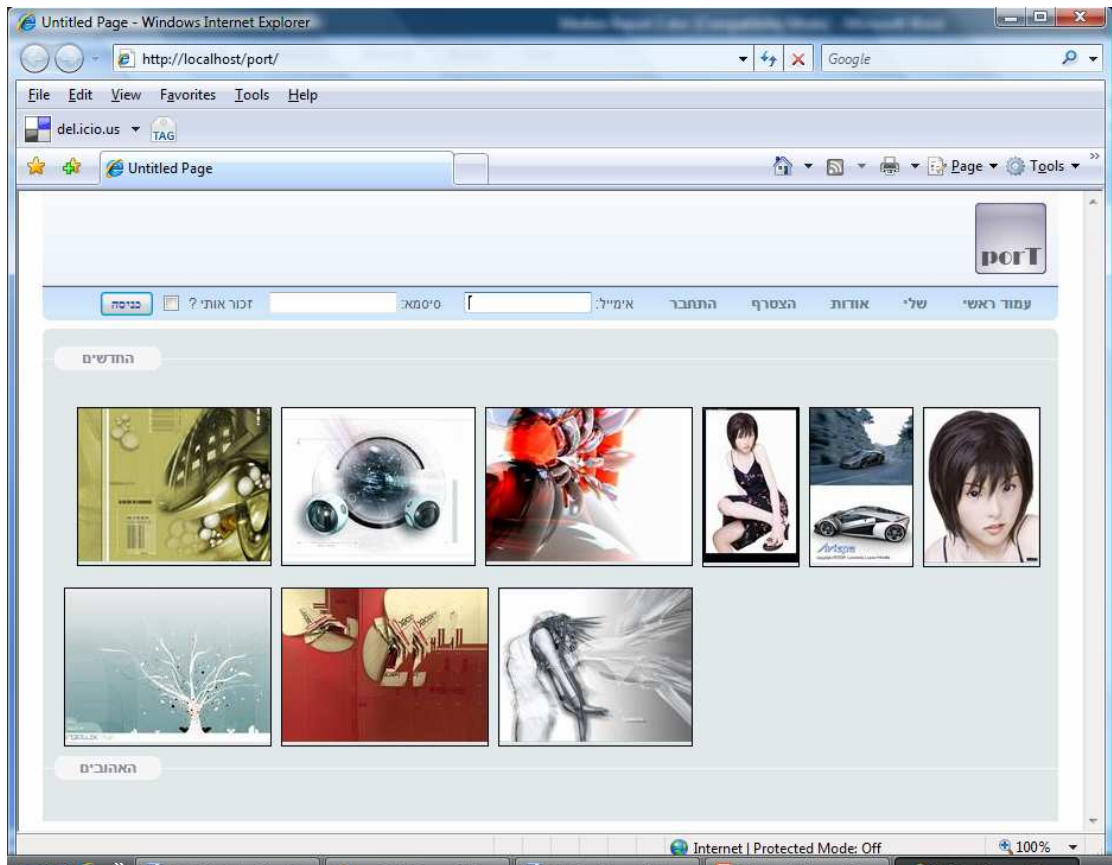
`ImageSizeConstraint` – StrictWidth, StrictHeight, or NoStrict

7.4.6.

4. User Interface Design

4.1. HomePage

This is the entry page of the web application, and should provide an overview of the newest and best items in the system.



The page allows logging in, navigation to other pages, and navigation to item pages for the displayed images.

4.2. Registration

This is the registration page

The screenshot shows a Windows Internet Explorer browser window displaying a registration page. The address bar shows the URL `http://localhost/port/register.aspx`. The page has a header with the 'port' logo and navigation links: 'עמוד ראשי' (Home), 'שלי' (My), 'אודות' (About), and 'התחבר' (Login). The main heading is 'הצטרף לקהילה' (Join the community). Below this, there are two columns of text. The left column contains a paragraph of text in Hebrew, followed by a registration form with four input fields and a 'רישום' (Register) button. The right column contains the heading 'פרטי משתמש' (User details) and a list of labels for the form fields: 'כתובת דואר אלקטרוני' (Email address), 'שם משתמש' (Username), 'סיסמא' (Password), and 'אישור סיסמא' (Confirm password).

הצטרף לקהילה

פרטי משתמש הסברים על המערכת

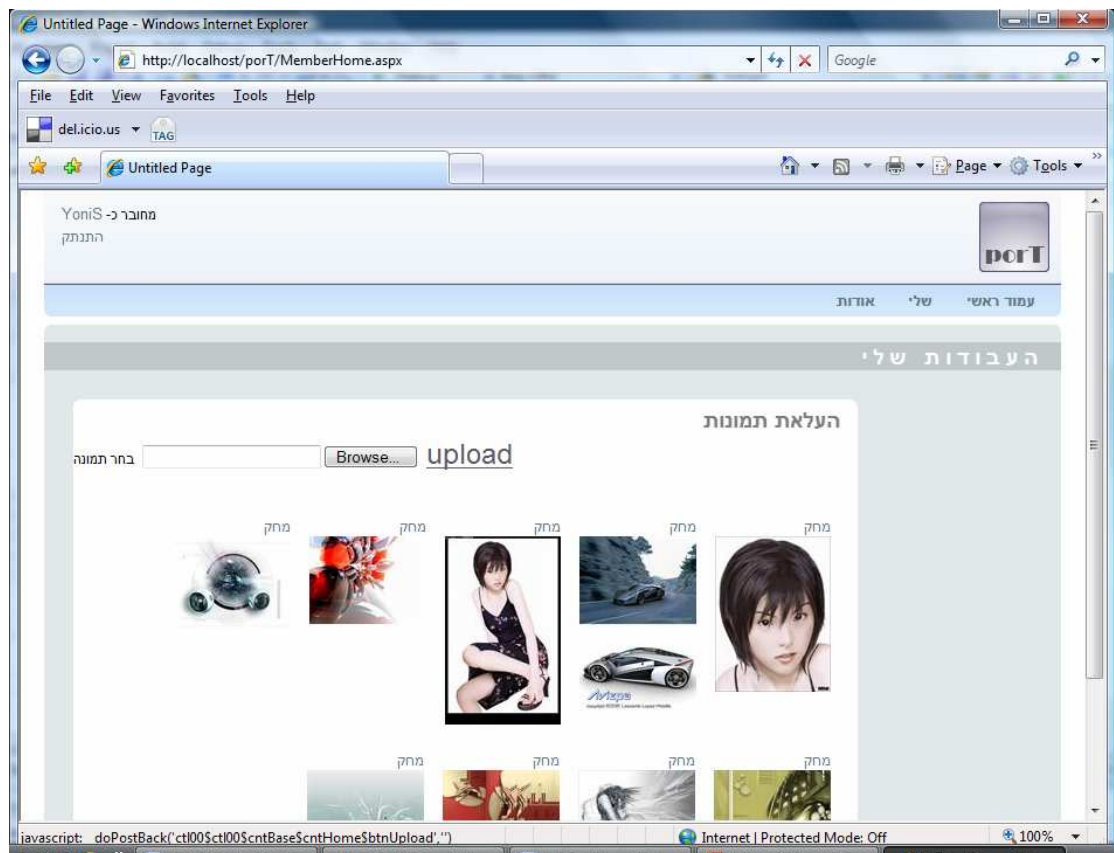
מבטיחים לא להציק - כתובת זו משמשת לזיהוי החשבון האישי שלכם בלבד.
אהמ, אנחנו מקווים שאתם עדיין זוכרים את השם שלכם... ואם אפשר - באנגלית
בבקשה.
סיסמת הכניסה לחשבון שלכם...
אם יודעים שזה מעצבן, אבל זה לטובתכם... באמת.

כתובת דואר אלקטרוני
שם משתמש
סיסמא
אישור סיסמא

רישום

4.3. Member Home

Member portfolio Management page



4.4. Login screen

This screen will be the fallback-page when login fails from home-page, or when there was an attempt to view a member-only section, before logging in.

It will contain the email and password text boxes, and a login button, in addition to registration navigation links, and other explanations about the system.

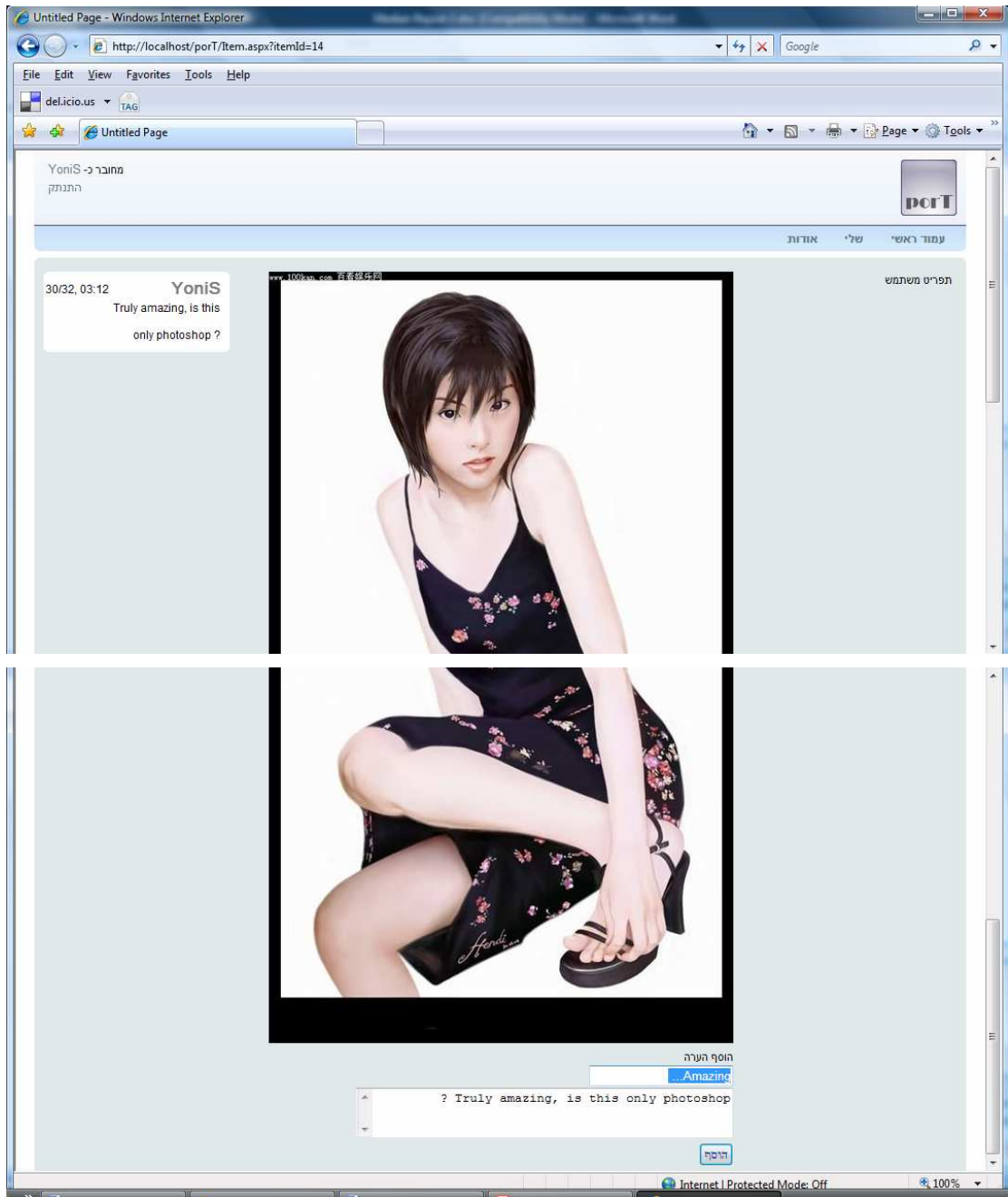
4.5. Community navigation page

This screen will allow filtering and advanced navigation across the newest and most popular items in the system.

It will allow navigation by categories, paging, filtering by dates and image sizes.

4.6. Item View page

This screen will show a specific item, and additional details concerning it, such as comments, rating, viewing statistics and artist notes.



The page also allows rating the image, and adding comments, if the user is logged in.

4.7. Additional Material

This section is not applicable.

9.3 . STD – Software Test Documentation - מסמך בדיקות

"porT" – community art portal – Software Test Plan

5. Introduction

This document will specify the different testing tasks to be performed at each project junction, and will elaborate on each task as to what functionality is tested, which component is tested and what are the expected test results.

6. Test items

The test items are divided into three major categories – Backend, business logic and user interface.

Backend and business logic tests will be carried out by a unit testing framework (NUnit), which will define coded tests, executing on each module or entity, as will be described in the following.

7. Features to be tested

Backend tests

1. Entity persistence test –

- a. Will be carried out on all entities in the system
- b. Will start with the least dependent entities, and escalate to the most dependent ones.
- c. Test flow :
 - i. Open a database session
 - ii. Create a new entity
 - iii. If an entity has relationship constraints – mock parents/children/kindred entities will be created as well.
 - iv. Save the entity to the database
 - v. Close the database session
 - vi. Open a new database session
 - vii. Getting the entity from the database by ID
 - viii. Verify that all properties and related entities exist and have been persisted successfully. (compare to creation values)
 - ix. Close the database session and Open a new one
 - x. Get the entity from the database by ID
 - xi. Delete the entity.
 - xii. Try to get the entity by ID – verify that the returned object is null. (entity cache updated successfully)
 - xiii. Close the database session and open a new one
 - xiv. Try to get the entity by ID – verify that the returned object is null. (entity deleted successfully)
- d. Success is defines by whether all entities have been created, saved, queried, validated and deleted successfully.

- 2. Business logic tests
 - a. Forum will not be included in backend testing (3rd party component)
 - b. Test flow :
 - i. Initiation – Each test will create relevant mock instances relevant to the test.
 - ii. Execution – call the relevant business logic function
 - iii. Evaluation – compare execution results to expected results
 - c. Business logic tests will include all functions in all modules in the BL layer, and specifically :
 - i. Create a new user – registration.
 - 1. Verify that authentication is successful post registration
 - 2. Verify that the email is sent (relevant service function called)
 - 3. Verify user status is "not activated"
 - ii. Add an image to my portfolio
 - 1. Verify portfolio item counters are updated
 - 2. Verify image is accessible and retrievable
 - iii. Delete an image from my portfolio
 - 1. Verify portfolio item counters are updated
 - 2. Verify that community content and favorite relationships are updated.
 - iv. Item querying –
 - 1. Call GetItems() with all available permutations – filtering by category, popularity/recency sorting, public/private etc
 - 2. Verify results against expected results.
 - v. Vote
 - 1. Verify that related item rank is updated and persisted
 - 2. Verify that vote is persisted and a second vote is not allowed (uniqueness constraint)
 - vi. Save To Favorites
 - 1. Verify that the related item rank is updated and persisted
 - 2. Verify that save is persisted and item now appears in GetFavorites(User).
 - vii. Remove from favorites
 - 1. Verify that the related item rank is updated and persisted
 - 2. Verify that delete is persisted and item does not appear in GetFavourites(User).
 - viii. Adding a comment
 - 1. Verify comment is persisted and retrieved with the Item.Comments collection.
 - 2. Verify comment properties

3. Verify relationship to commenter

d. User interface tests –

- i. Manual tests, performed against the application itself (browser)
- ii. Will include all functionality depicted in the SRD
- iii. Input validation tests
- iv. Application security and access moderation tests - attempting to access unauthorized data or functionality – Penetration tests. (Sql injection input pattern tests, script injection/cross site scripting validation tests, privacy tests)

8. Features not to be tested –

Load tests will not be performed as part of the proof-of-concept, but should be further discussed as part of the production migration phase.

9. Environmental needs –

Tests will be carried out in two separate environments –

Backend tests will be performed against the UnitTest database, and frontend tests (complete user functionality) will be tested against a QA database.

The frontend test server environment (QASRV) will run on a dedicated virtual machine, and include an application version of the latest complete iteration, which will be associated with the QA db instance.

The frontend test client environment (QACLI) testing will be performed from a dedicated virtual machine, in order to negate the possibility of unknown client dependency on an installed component from the development environment, and to simulate a real client-server behavior.

10. Schedule

All backend testing tasks will be performed completely on each code iteration (about every 3 days), and each new backend component or entity will be tested specifically upon creation. (the test code should be written before the tested code !)

This will be performed as part of articles 19 and 32 in SPMP schedule Gantt chart.

Frontend testing will be performed upon each iteration completion, on the following basis:

1. Basic functionality will be retested (Registration, login, and verifying all page views load successfully)
2. Iteration specific functionality will be tested.

Upon project completion, a complete backend, BL, and frontend test will be the project acceptance test (article 33 in SPMP schedule Gantt chart).

9.4. דוח בדיקות תוכנה – Software Test Report

Purpose	Inputs	Expected outputs	Test Procedure	Test Result	Pass/Fail
Entity persistence test (Automated)	List of entity types	data in DB	Create an entity of each type, fill up with data and save to DB	none	pass
Registration	User details	Successful registration, authentication and redirection to member home page	Click the registration link, fill up valid details and click "register"	redirected and authenticated	pass
Add image to portfolio	Item details + file	Successful upload, redirection to member home page, with the new item displayed in the list	Click "add item" link, fill up valid details and click "upload"	redirected and item appears	pass
Delete image from portfolio	item	item deleted from list, counters updated	go to member home and delete an existing item	item deleted, counters updated	pass
Item Querying	filter object	list of items which answer filter criteria	go to navigation screen and play with filter options, verify that results are consistent with what's in the DB	result list consistent with db data	pass
Voting	item, user	vote is saved to database, counters updated	Login, cast a vote on the item through the floating menu.	vote is saved to database, counters updated	pass
Save to favorites	item, user	record is saved to database, counters updated, item appears in users favorites	Login, save the item to your favorites using the hover menu	item saved, appears in favs.	pass
Remove from favorites	item, user	relation is deleted from database, counters updated, item no longer appears in users favorites	Login, go to your favorites, and remove an item using the floating menu	item removed from favs, record deleted	pass
Adding a comment	item, user, text	comment added to db, displayed in page	Login, click on an item, add a comment	comment added	pass
Authorization test	item	item won't be deleted	record a "delete" request for your item, and modify the http request to address someone else's item	delete failed	pass
Validation	none	friendly error displayed	Try submitting item/comment without filling some fields	friendly error displayed	pass

9.5 מסמך ניהול הפרוייקט - SPMP – Software Project Management Plan

"porT" – community art portal – Software Project Management Plan

1. Introduction

This document contains project management details relevant to the "porT" project - a final project produced as part of the Academic College of Engineering B.Sc. program in software engineering.

2. Risk Management

No changes have been introduced to the risk assessment or planned schedule since the project initiation document.

a. Hardware failure on the development/production system

- a. Probability – low/medium
- b. Precautions – setup of a redundancy cluster for system servers and relevant infrastructure
- c. Project influence – est. delay of two weeks

b. Database data loss

- a. Probability – medium
- b. Precaution/ Recovery – Weekly full backups and daily incremental backups should be performed on all database instances
- c. Project influence – est. delay of one week, in case of recovery failure

c. Data loss in code base

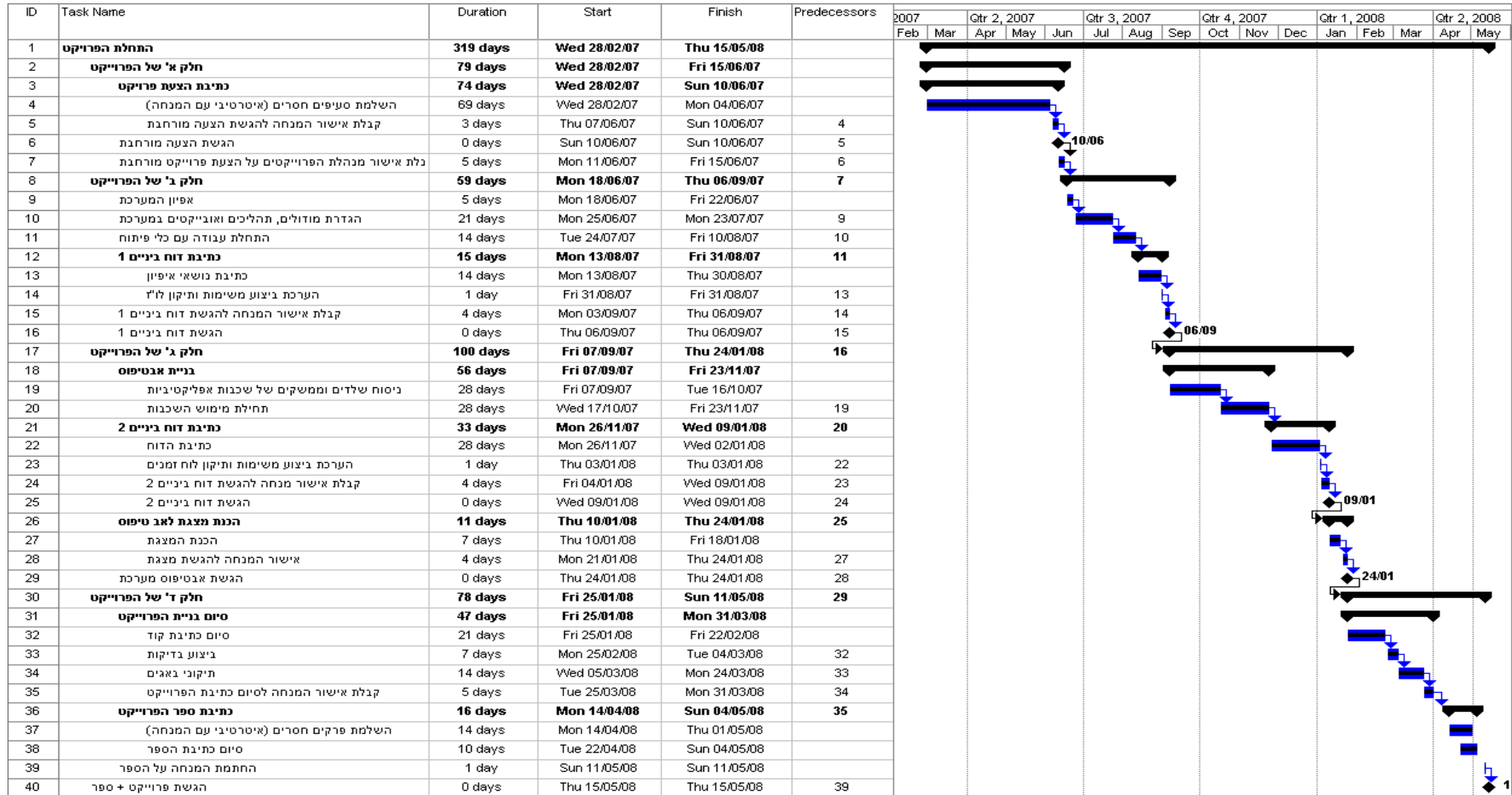
- a. Probability – low/medium
- b. Precautions – backups of the code base should be performed on each iteration completion
- c. Project influence – depends on project progress. Estimated one day per week of progress (starting development phase)

3. Evaluation of the SPMP

The project is currently on schedule, current status is – interim report A has been sent to supervisor on the 1st of September, two days earlier than planned – to allow remarks and comments to be addressed appropriately.

The next stage in the project is the creation of a prototype system, which will provide the basis for development and exemplify the ideas and feature rollout, which are currently in concept.

4. Schedule



מחלקת הנדסת תוכנה

שם הפרויקט: פורטל אומנות דיגיטלית
Project Name: Digital Art Portal

הצעת פרויקט

שם הסטודנט: שלום יוני

מספר תעודת זהות: 0-3909419-8

שם המנחה: יונית שוורץ וורבר

תאריך ההגשה: 16/5/2007

תוכן עניינים

90	תמצית	1.
91	מטרות ומדדים	2.
91	2.1 היעד הסופי ומדד הצלחה מעשי	
91	2.2 מדדי הצלחה (תיאורטי)	
92	3. דרישות ותיחום	
94	4. תכנית בדיקות ראשונית	
95	5. אופן הביצוע	
96	6. אמצעים/ כלים	
97	7. ניתוח חלופות	
98	8. ניתוח פונקציונלי ראשוני	
98	9. ניתוח סיכונים	
99	10. פערי ידע	
99	11. תוצרי הפרוייקט	
	99 WBS	12.
100	תרשים Gantt	13.
	רשימת מקורות	14.
	שגיאה! הסימניה אינה מוגדרת.	

1. תמצית

הפרוייקט המוצע הינו פורטל אומנות דיגיטלית אשר מטרתו העיקרית לתת במה לאומנים להציג את כישוריהם בפני הקהילה ולקבל משוב על יצירותיהם. הפורטל יאפשר לחבריו ליצור "תיק עבודות" דיגיטלי, לשוחח על יצירות של חברים אחרים ולנקד את היצירות.

קהל היעד העיקרי של הפורטל הוא סטודנטים לעיצוב חזותי, "ביצועיסטים" בתחום, חובבי צילום ומתעניינים.

קיימים מספר אתרים בתחום, כגון DeviantArt.com העולמי, ו- stage.co.il / cgtalk.co.il הישראליים.

מטרת הפורטל היא לתת פתרון ידידותי יותר למשתמש, יותר ממוקד בקהילה הישראלית, בדגש על יצירות ברמה גבוהה.

מטרה נוספת הינה להציג את יכולותיהם של מועמדים למגמות העיצוב החזותי במוסדות להשכלה גבוהה בארץ, ולאפשר להם לפרסם את יצירותיהם ולקבל משוב מהקהילה כחלק מתהליך בניית תיק העבודות שלהם.

מפתח המערכת : יוני שלום

מנחת הפרוייקט : יונית שוורץ וורבר

2. מטרות ומדדים

כיום קיימות מספר מערכות דומות, כגון DeviantArt.com העולמית, ו-stage.co.il או cgart.co.il הישראליות. המטרה העיקרית של המערכת המוצעת הינה להביא לתחום יכולות ופונקציונליות נוספת אשר תאפשר למנף את יכולות המערכת לטובת קידום האישי בתחום, בין אם לצורך שיפור תיק העבודות לקראת קבלה למוסד לימודים, או לצורך התמודדות על משרה. הצרכים העיקריים עליהם באה לענות המערכת המוצעת הינם :

- מתן דגש על פיתוח אישי של היוצר
- שיפור ממשק המשתמש
- הצעת יכולות חדשות שלא קיימות באתרים הנוכחיים, והכנסת אלמנטים "תחרותיים" המערכת הינה אתר אינטרנט המקושר לתחום הפנאי והבידור, וככזה אינו בא לענות לצורך ארגוני או ניהולי כלשהו. המערכת תוערך על הצלחתה על פי מדדים של רישום משתמשים ביחס לחשיפה, ועל פי כמות הפעילות השוטפת באתר.

7.4.7.2.1 היעד הסופי ומדד הצלחה מעשי

- הקמת מערכת פעילה הכוללת את הפונקציונליות המתוארת בדרישות.

7.4.8.2.2 מדדי הצלחה (תיאורטי)

- אחוז הנרשמים מתוך המשתמשים שנחשפו לאתר. (מעל 20%)
- אחוז המשתמשים החוזרים פעם שניה, מתוך אלו שנרשמו. (מעל 40%)
- אחוז המשתמשים הפעילים באתר (2 כניסות יחודיות) מתוך אלו שנרשמו. (מעל 1000 כניסות יחודיות ביום)
- כמות הכניסות היומית לאתר ביחס לגודל קהל היעד המוערך (מעל 40% מקהל היעד)

(הערה : עקב העובדה שהאתר לא יהיה חשוף לאינטרנט במסגרת הקמת הפרוייקט ולא יבוצעו פעילויות שיווק, מדדי הצלחה אלו הינם תיאורטיים בלבד).

3. דרישות ותיחום

- תחום העבודה - Web Community
- תחום המוצר - Digital Art
- דרישות מידע ופונקציונליות
 - יצירת חשבון אישי – אימייל וסיסמה + תמיכה ב Net CardSpace.
 - ניהול החשבון האישי
 - איחסון והצגה של תמונות לפי קטגוריות
 - שמירת "הערות היוצר" על היצירה
 - הצגה של תמונות בפורמטים שונים ובגדלים שונים
 - יצירת תיק עבודות אישי ובהצגה פומבית של תיק עבודות זה
 - דירוג יצירות ומתן ניקוד ליוצר
 - כתיבת Comments על יצירות
 - פורום
 - "יצירה בהמשכים" – שיתוף פעולה של מספר יוצרים על אותה יצירה
 - סימון "יצירות אהובות" לתוך תיקיית "מועדפים" אישית
 - נתון "סימנו אותך במועדפים" עבור משתמש ועבור יצירות ספציפיות
 - שיקלול דירוגים עבור יוצרים והצגת "היוצרים האהובים" של השבוע/ החודש/ השנה
 - דירוג של יצירה מושפע באופן שונה ממי שיש לו הרבה נקודות וממי שיש לו פחות
- ארכיטקטורה
 - פיתוח במודל השכבות – שכבת DAL, EL, BL
 - מודולים
 - Images – איחסון שליפה והצגה של תמונות
 - Portfolio – ניהול תיק עבודות
 - Statistics – כמה אהבו, דירוג ממוצע של מי שאהב, דירוג היצירה וכו'
 - ItemServices – ניהול המידע על יצירת אומנות מסויימת
 - UserStore – מידע על משתמשים
 - Admin – ממשק ניהול לאתר

שכבות אפליקטיביות עיקריות

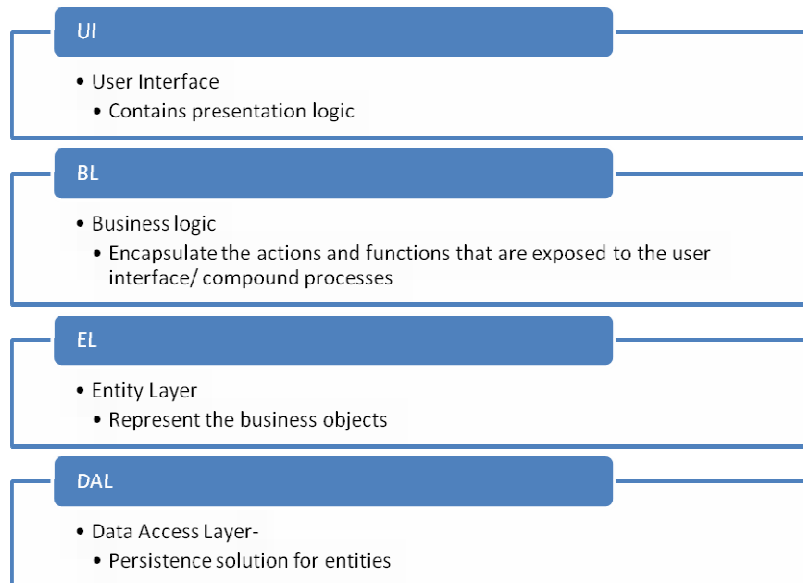


Figure 3 - שכבות אפליקטיביות

רכיבי BL עיקריים :

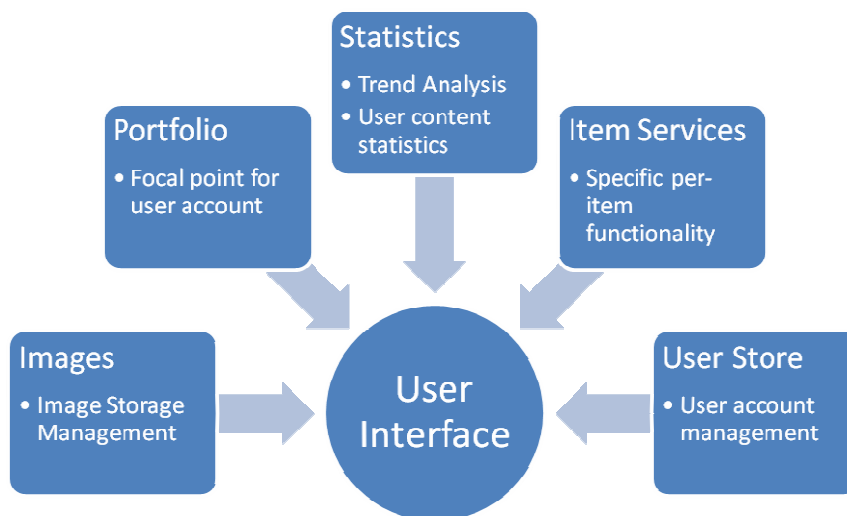


Figure 4 - רכיבי שכבת שירות/ BL

4. תכנית בדיקות ראשונית

- דרישות מקדימות לבדיקות :

- בדיקה באמצעות מחשב מרוחק, על גבי דפדפנים שונים (Mozilla FireFox ,IE6/7)
- איכלוס מסד הנתונים של המערכת במספר חשבונות משתמש ופריטי מידע, לצורך סימולציה של פעילות שוטפת - יתבצע באמצעות DB Scripts או Unit Test ייעודי, מראש.

- פונקציונליות נבדקת :

- כניסה לאתר וצפייה בעמוד הבית
- רישום לאתר - יצירת חשבון משתמש
- עריכת פרטים אישיים
- צפייה בתיק העבודות האישי
- צפייה ביצירה מסויימת ועריכתה
- הוספת יצירה
- הוספת משוב/ דירוג
- הצגת שיקלול הדירוגים ליצירה
- הצגת שיקלול דירוגים ליוצר
- סימון וצפייה באוסף יצירות אהובות
- צפייה "יצירות פופולריות" לפי כמות צפיות, דירוג משוקלל או דירוג יוצר
- צפייה ביצירות לפי קטגוריות
- צפייה בפרטי יוצר

5. אופן הביצוע

- ביצוע מחקר ראשוני של האתרים הקיימים בתחום
 - טכנולוגיות בשימוש
 - ממשקי משתמש
 - חשיפה וקהל משתמשים
- הגדרת מאפייני המערכת העיקריים - יכולות, פונקציונליות, ודרישות ממשק המשתמש
 - ניסוח הפונקציונליות וחלוקה לקטגוריות
 - ניסוח דרישות ממשק המשתמש
 - יצירת "Mockups" של ממשק המשתמש לחיקוי בעת יישום המערכת
 - ניסוח של מספר use cases
 - הגשת דו"ח ביניים 1 (ספטמבר 2007)
- ביצוע עיצוב גרפי של המערכת – מתן קווים מנחים ו"קונספט" ראשוני לממשק המשתמש.
 - הגדרת חלוקת רכיבי התצוגה
 - אפיון רכיבי תצוגה עיקריים כגון תפריטים, מעברים, רשימות וכו'
- תיכון המערכת
 - חלוקה לרכיבים פונקציונליים
 - ביצוע Architectural Design
 - ביצוע Detail design לכל רכיב ופונקציונליות כחלק מתהליך איטרטיבי (Agile)
 - הגשת דו"ח ביניים 2 (ינואר 2008)
- פיתוח המערכת והצגת אבטיפוס
 - בניית המודל הטבלאי של מסד הנתונים (Sql Server 2005)
 - כתיבת הקוד - ASP.Net/ C#
 - ביצוע הבדיקות משולב בתהליך הפיתוח, כחלק מהתהליך האיטרטיבי (TDD + Agile)
- בדיקת מערכת - השוואת תוצרים אל מול דרישות
- שיפורים ותחזוקה
- הגשת דו"ח סיום (מרץ 2008)

6. אמצעים/ כלים

סביבת הפיתוח -

- סביבת פיתוח – Visual Studio 2005
- שרת Web – IIS 6.0
- שרת Sql Server 2005
- מחשב פיתוח עיקרי :
 - זיכרון 2GB
 - מעבד Athlon 3800x2
- מחשב פיתוח משני :
 - IBM Thinkpad

סביבת ריצה –

- שרת Web – IIS 6.0
- שרת Sql Server 2005
- פתרון Hosting -
 - שרת Proliant 360D - Proliant 360D ,Dual Xeon3.0 ,4GB Ram
- רכישת שם הדומיין www.porT.co.il

סביבת משתמש –

- מחשב עם גישה לאינטרנט
- דפדפן

7. ניתוח חלופות

- החלופות למערכת הינן אתרים דומים בתחום כגון DeviantArt.com העולמי, ו- stage.co.il /cgtalk.co.il הישראליים.
 - DeviantArt.com - אתר עולמי אשר עוסק בתחום של אומנות דיגיטלית, מהווה אחד מ"אבני היסוד" של הקהילה העולמית. מכיל פונקציונליות רבה, אך הינו כללי מאוד ומהווה במה לכל סוג יצירה אפשרי, מכל האסכולות. קהל היעד של האתר הינו רחב, ומכיל חובבי אומנות דיגיטלית ומקצוענים כאחד.
 - cgTalk - פורום ישראלי אשר מאפשר גם הצגה ופרסום של יצירות. האתר פיתח לעצמו נישא מקצועית יחסית, וקהל המשתמשים שלו כולל מקצוענים וסטודנטים בתחום, והוא פונה פחות אל ה"גולש המזדמן".
 - Stage.co.il - אתר אשר פונה יותר אל ה"גולש המזדמן", אינו מתמקד בנישה מסוימת ומכיל יצירות ברמה יחסית נמוכה יחסית לאחרים. היצירות באתר זה אינן יצירות אומנות דיגיטלית בלבד, אלא גם יצירות אומנות מתחום השירה, הציור והאנימציה.
- חלופות עיקריות לטכנולוגיית הפיתוח הן Asp, PHP, JSP, Ruby on Rails, ASP.Net, MonoRail. הבחירה ב- ASP.Net היא מהסיבות הבאות:
 - מיומנות קיימת של המפתח בטכנולוגיה
 - יכולות Object Oriented והתאמה למתודולוגית Test Driven ,
 - באמצעות כלים מובנים כגון NUnit.
 - יכולות Ajax קלות להטמעה בשימוש ASP.Net Ajax - חיסכון בזמן פיתוח.
 - יכולות ORM באמצעות NHibernate - Net ORM Framework.
- חלופות עיקריות ל-DB הינן MySQL ו- Oracle. הבחירה ב- SQL Server 2005 מהסיבות הבאות:
 - מיומנות קיימת של המפתח במסד הנתונים
 - רמת תאימות גבוהה ל- NHibernate ORM Framework
 - מאפשר הטמעה קלה ושימוש ב- Cache Dependency של .Net ,
 - באמצעות Polling או SQL Server Agent.

8. ניתוח פונקציונלי ראשוני

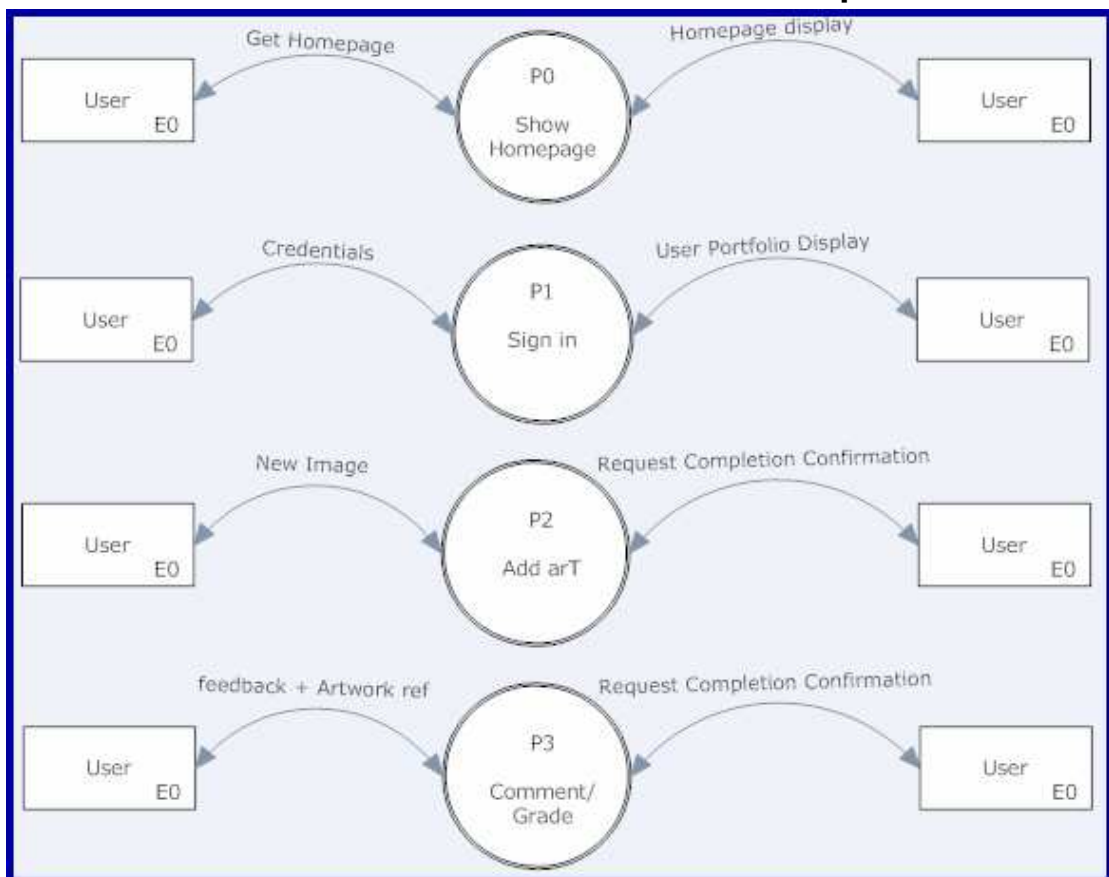


Figure 5 - ניתוח פונקציונלי

שאר הפונקציונליות תיבנה כחלק מתהליך איטרטיבי אשר בו יקבעו הדרישות הספציפיות בצמוד לתחילת איטרציית המימוש הבאה.

9. ניתוח סיכונים

- כשלי חומרה בסביבות הפיתוח/ הריצה
 - סבירות - נמוכה/ בינונית
 - מניעה - יציאת Redundancy ומנגנוני Failover עבור שרתי המערכת והתשתיות הרלוונטיות.
 - השפעה על הפרויקט - עיכוב משוער של כשבועיים
- איבוד מידע במסד הנתונים בסביבות הפיתוח/ הריצה
 - סבירות - בינונית
 - מניעה/ התאוששות - ביצוע גיבויים רציפים כל גירסה, ובסביבת הריצה - גיבוי יומי (אינקרמנטלי) ושבועי (מלא).
 - השפעה על הפרויקט - עיכוב משוער של כשבועיים, תלוי בסטטוס התקדמות

- אובדן מידע ב- Code Base
 - סבירות - נמוכה/ בינונית
 - מניעה/ התאוששות - ביצוע גיבוי קוד המערכת בכל גרסה (כאחת לשבועיים)
 - השפעה על הפרוייקט - תלוי בכמות המידע שאבד/ סטטוס התקדמות.

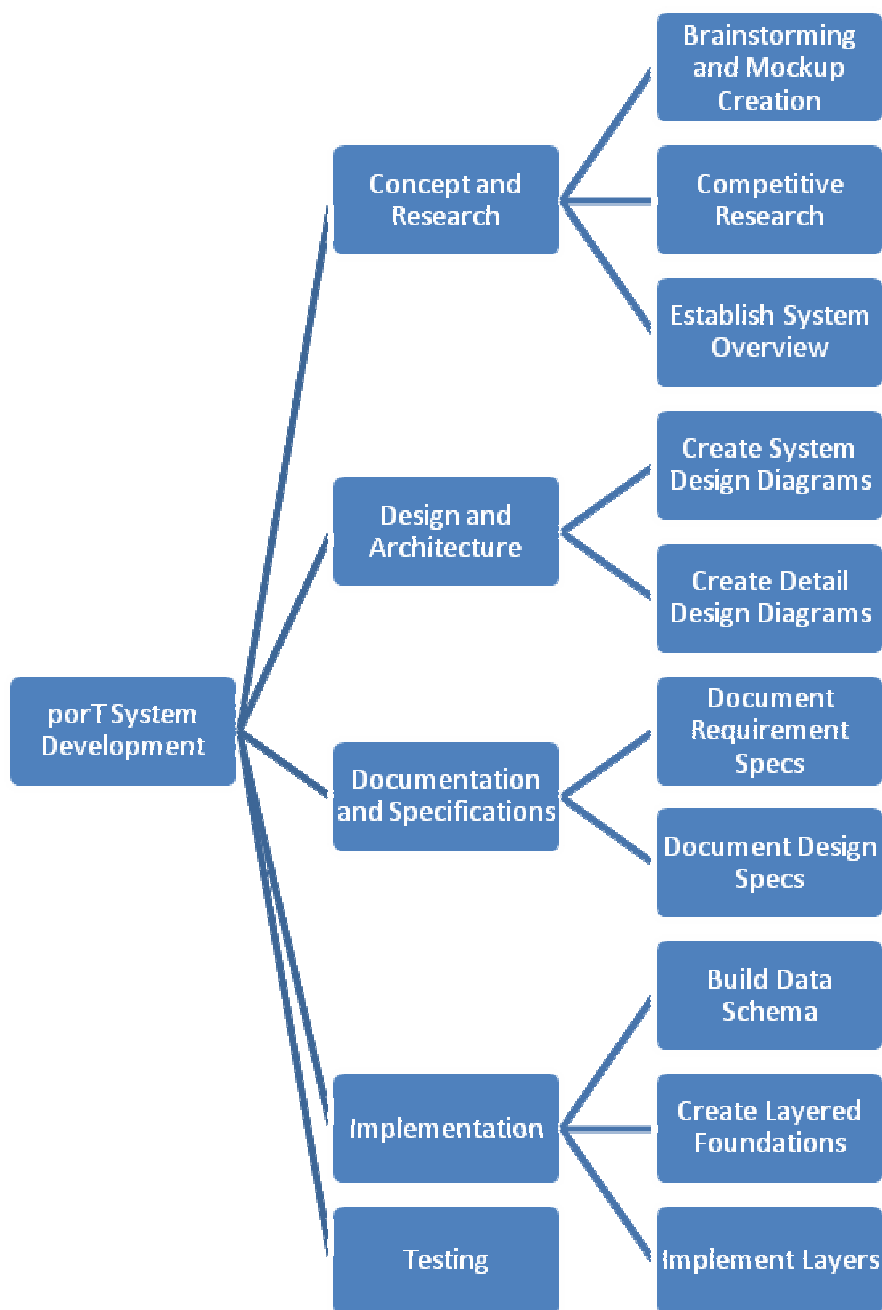
10. פערי ידע

אין פערי ידע להשלים.

11. תוצרי הפרוייקט

- דו"ח 1, דו"ח 2
- אבטיפוס מערכת porT
- מצגת הפרוייקט
- דו"ח מסכם
- מערכת porT הסופית

12. WBS



13. תרשים Gantt

